Reparto de tráfico con routers CISCO

Pablo Collado Soto

Ingeniería de Tráfico

1. Introducción

En esta práctica estudiaremos los mecanismos de reparto de tráfico de los routers CISCO. En una primera instancia lo haremos con equipos virtuales para luego generar el tráfico con las máquinas virtuales tal y como veníamos haciendo en las prácticas anteriores.

2. Pruebas con equipos virtuales

Tras realizar la configuración que se explica en el guión de la práctica llegamos a una topología como la que aparece en la figura 1. Además de configurar las direcciones IP así como el protocolo de encaminamiento OSPF hemos activado el reparto de tráfico en R1. Dado que todo el tráfico es generado por el PC1 solo debemos configurar este reparto de tráfico en R1 porque en R2 todo lo que llegue saldrá por la interfaz e1/0.

2.1. Reparto de tráfico por destino (PD)

Al configurar el reparto de tráfico éste será *por destino* por defecto. Para poder juzgar si este reparto de tráfico está funcionando correctamente empezaremos por mostrar las estadísticas de paquetes emitidos por cada una de las "cubetas" del router. Esperamos observar que todas las cubetas estén a 0 en un inicio como se observa en la figura **??**. Después mandaremos una serie de **ping**s desde el PC1 al PC2 y observaremos que los 5 paquetes son asociados a una de las "cubetas" en función del **hash** del destino tal y como se comenta en el guión. Si **ping**eáramos la misma dirección de nuevo observaríamos que los paquetes salen por la misma cubeta al ser el destino idéntico. Podemos observar ésto en la figura 3. Si después mandamos una serie de **ping**s a la dirección 10.0.3.1, la interfaz del router que pertenece a la subred asociada al PC2 estaremos cambiadno el destino, con lo que esperamos que los paquetes salgan por la otra interfaz con mismo coste a la anterior. Los primeros **ping**s salen por la interfaz FastEthernet0/1 asociada a la dirección 10.0.1.2 como se observa en la figura 3 mientras que los siguientes salen por la interfaz FastEthernet0/0 asociada a la dirección 10.0.2.2 como apreciamos en la figura 4.

Con esto concluimos que el reparto de tráfico por rutas de igual coste basándonos en el destino de los paquetes funciona correctamente. Como curiosidad señalamos que el primer par de **ping**s se

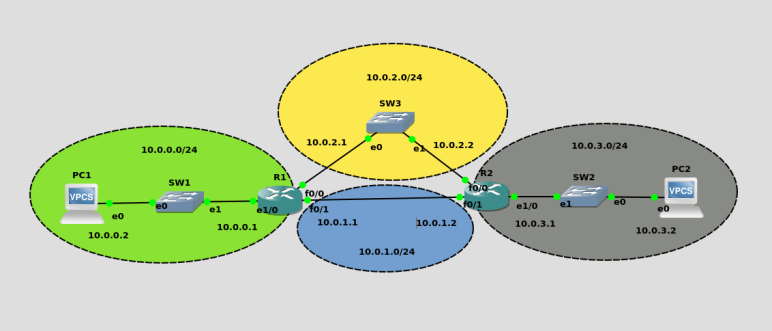

Figura 1: Topología bajo estudio en la primera parte.

```
R1#sh ip cef 10.0.3.0 internal
10.0.3.0/24, version 28, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.0.2.2, FastEthernet0/0, 0 dependencies
    traffic share 1
    next hop 10.0.2.2, FastEthernet0/0
    valid adjacency
  via 10.0.1.2, FastEthernet0/1, 0 dependencies
    traffic share 1
    next hop 10.0.1.2, FastEthernet0/1
    valid adjacency

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
           internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface                 Address         Packets
  1     Y   FastEthernet0/0           10.0.2.2              0
  2     Y   FastEthernet0/1           10.0.1.2              0
  3     Y   FastEthernet0/0           10.0.2.2              0
  4     Y   FastEthernet0/1           10.0.1.2              0
  5     Y   FastEthernet0/0           10.0.2.2              0
  6     Y   FastEthernet0/1           10.0.1.2              0
  7     Y   FastEthernet0/0           10.0.2.2              0
  8     Y   FastEthernet0/1           10.0.1.2              0
  9     Y   FastEthernet0/0           10.0.2.2              0
  10    Y   FastEthernet0/1           10.0.1.2              0
  11    Y   FastEthernet0/0           10.0.2.2              0
  12    Y   FastEthernet0/1           10.0.1.2              0
  13    Y   FastEthernet0/0           10.0.2.2              0
  14    Y   FastEthernet0/1           10.0.1.2              0
  15    Y   FastEthernet0/0           10.0.2.2              0
  16    Y   FastEthernet0/1           10.0.1.2              0
  refcount 6
```

Figura 2: Estadísticas de paquetes antes de mandar ningún `ping`.

```
R1#sh ip cef 10.0.3.0 internal
10.0.3.0/24, version 28, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.0.2.2, FastEthernet0/0, 0 dependencies
    traffic share 1
    next hop 10.0.2.2, FastEthernet0/0
    valid adjacency
  via 10.0.1.2, FastEthernet0/1, 0 dependencies
    traffic share 1
    next hop 10.0.1.2, FastEthernet0/1
    valid adjacency

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
           internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface                 Address         Packets
  1     Y   FastEthernet0/0           10.0.2.2              0
  2     Y   FastEthernet0/1           10.0.1.2              0
  3     Y   FastEthernet0/0           10.0.2.2              0
  4     Y   FastEthernet0/1           10.0.1.2              0
  5     Y   FastEthernet0/0           10.0.2.2              0
  6     Y   FastEthernet0/1           10.0.1.2              0
  7     Y   FastEthernet0/0           10.0.2.2              0
  8     Y   FastEthernet0/1           10.0.1.2              5
  9     Y   FastEthernet0/0           10.0.2.2              0
  10    Y   FastEthernet0/1           10.0.1.2              0
  11    Y   FastEthernet0/0           10.0.2.2              0
  12    Y   FastEthernet0/1           10.0.1.2              0
  13    Y   FastEthernet0/0           10.0.2.2              0
  14    Y   FastEthernet0/1           10.0.1.2              0
  15    Y   FastEthernet0/0           10.0.2.2              0
  16    Y   FastEthernet0/1           10.0.1.2              0
  refcount 6
```

Figura 3: Estadísticas de paquetes tras los primeros 5 `ping`s.

```
R1#sh ip cef 10.0.3.0 internal
10.0.3.0/24, version 28, epoch 0, per-destination sharing
0 packets, 0 bytes
  via 10.0.2.2, FastEthernet0/0, 0 dependencies
    traffic share 1
    next hop 10.0.2.2, FastEthernet0/0
    valid adjacency
  via 10.0.1.2, FastEthernet0/1, 0 dependencies
    traffic share 1
    next hop 10.0.1.2, FastEthernet0/1
    valid adjacency

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
           internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface                 Address         Packets
  1     Y   FastEthernet0/0           10.0.2.2              5
  2     Y   FastEthernet0/1           10.0.1.2              0
  3     Y   FastEthernet0/0           10.0.2.2              0
  4     Y   FastEthernet0/1           10.0.1.2              0
  5     Y   FastEthernet0/0           10.0.2.2              0
  6     Y   FastEthernet0/1           10.0.1.2              0
  7     Y   FastEthernet0/0           10.0.2.2              0
  8     Y   FastEthernet0/1           10.0.1.2              5
  9     Y   FastEthernet0/0           10.0.2.2              0
  10    Y   FastEthernet0/1           10.0.1.2              0
  11    Y   FastEthernet0/0           10.0.2.2              0
  12    Y   FastEthernet0/1           10.0.1.2              0
  13    Y   FastEthernet0/0           10.0.2.2              0
  14    Y   FastEthernet0/1           10.0.1.2              0
  15    Y   FastEthernet0/0           10.0.2.2              0
  16    Y   FastEthernet0/1           10.0.1.2              0
  refcount 6
```

Figura 4: Estadísticas de paquetes tras los segundos 5 `ping`s.

pierden por un *timeout* debido a que al estar las tablas de `ARP` vacías se deben popular en toda la red, proceso que introduce una latencia.

2.2. Reparto de tráfico por paquetes (PP)

Para configurar el reparto de tráfico por paquetes debemos simplemente especificarlo en la terminal de configuración para sobrescribir el comportamiento por destino que se aplica por defecto. El estado de las estadísticas de paquetes tras aplicar este nuevo reparto de tráfico sigue siendo el que encontrábamos en la figura 4. Tras volver a `ping`ear al `PC2` observamos las estadísticas de paquetes de nuevo y llegamos al resultado de la figura 5. Debemos recordar que, sin opciones adicionales, enviamos 5 `ping`s con lo que veremos que aparecen 5 paquetes adicionales en las estadísticas. Como ahora el reparto tiene una granularidad a nivel de paquete asistimos a unas estadísitcas que envían cada paquete por una de las interfaces de manera alterna. El primero de todos va asociado al `hash` 1 y sale por la interfaz `FastEthernet0/0` y el quinto, asociado al `hash` 5, es emitido también por la interfaz `FastEthernet0/0`. Vemos que la interfaz de salida se alterna con cada paquete. Nótese que el primer contador es 6 porque partimos del escenario anterior mostrado en la figura 4. De todo esto concluimos que el reparto por paquetes también funciona.

También hemos realizado las pruebas propuestas ejecutando un `trace 10.0.3.2` desde `PC1` y viendo que la dirección `IP` del segundo salto varía entre `10.0.1.2` y `10.0.2.2` tal y como cabría esperar con el balanceo que hemos configurado. Con el balanceo por destino este salto no cambiaría porque el destino de los paquetes emitidos sería siempre el mismo...

2.3. Deshabilitando el reparto de tráfico

Al deshabilitar el reparto de tráfico perdemos la capacidad de ejecutar el comando `show ip cef 10.0.3.0 internal` con lo que dependemos de la información mostrada por `show interfaces stats`. Debemos recordar que en las estadísticas mostradas por este comando tenemos "ruido" de fondo introducido por protocolos como `OSPF` que envían mensajes de manera periódica. Ésto supone que si ejecutamos este comando en dos instantes distintos veremos cómo la los contadores de paquetes emiti-

```
R1#sh ip cef 10.0.3.0 internal
10.0.3.0/24, version 28, epoch 0, per-packet sharing
0 packets, 0 bytes
  via 10.0.2.2, FastEthernet0/0, 0 dependencies
    traffic share 1
    next hop 10.0.2.2, FastEthernet0/0
    valid adjacency
  via 10.0.1.2, FastEthernet0/1, 0 dependencies
    traffic share 1, current path
    next hop 10.0.1.2, FastEthernet0/1
    valid adjacency

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
           internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface                 Address         Packets
  1     Y   FastEthernet0/0           10.0.2.2              6
  2     Y   FastEthernet0/1           10.0.1.2              1
  3     Y   FastEthernet0/0           10.0.2.2              1
  4     Y   FastEthernet0/1           10.0.1.2              1
  5     Y   FastEthernet0/0           10.0.2.2              1
  6     Y   FastEthernet0/1           10.0.1.2              0
  7     Y   FastEthernet0/0           10.0.2.2              0
  8     Y   FastEthernet0/1           10.0.1.2              5
  9     Y   FastEthernet0/0           10.0.2.2              0
  10    Y   FastEthernet0/1           10.0.1.2              0
  11    Y   FastEthernet0/0           10.0.2.2              0
  12    Y   FastEthernet0/1           10.0.1.2              0
  13    Y   FastEthernet0/0           10.0.2.2              0
  14    Y   FastEthernet0/1           10.0.1.2              0
  15    Y   FastEthernet0/0           10.0.2.2              0
  16    Y   FastEthernet0/1           10.0.1.2              0
  refcount 6
```

Figura 5: Estadísticas de paquetes tras los primeros 10 `ping`s.

dos por interfaz crecen a un ritmo lento pero positivo aún sin haber emitido ningún tráfico de usuario desde el `PC1`. Para poder observar de manera más clara lo que ocurre con el tráfico de usuario modificaremos la invocación de `ping` para aumentar el número de mensajes y la cadencia con la que estos se emiten para así esclarecer las estadísticas que podemos obtener.

Para conseguir nuestro objetivo ejecutaremos el comando `ping 10.0.3.2 -c 500 -i 1` con lo que enviaremos 500 pings a una cadencia de 1 ms. El comportamiento que observamos es que todo el tráfico sale por una interfaz determinada. Incluimos la tabla de encaminamiento de `R1` en la figura **??** en la que se observa que la primera ruta al destino sale por la interfaz `FastEthernet0/0` asociada a la dirección `10.0.2.2` Ésta es la interfaz cuyo contador de paquetes emitidos se dispara tras someterla a una prueba de estrés con el comando `ping` modificado. De estos resultados deducimos que todo el tráfico se envía por la interfaz asociada a la primera ruta en la tabla de encaminamiento de `R1`. La elección de la interfaz podría ser aleatoria (desconocemos la implementación del sistema de los routers `CISCO`) pero creemos que como decíamos se escoge la primera entrada de la tabla de encaminamiento. En cualquier caso, si comparamos el estado inicial de la figura 6 con el final de la figura 7 nos daremos cuenta de que el contador de la interfaz `FastEthernet0/0` ha crecido muchísimo más que el de la interfaz `FastEthernet0/1` con lo que deducimos que, en ausencia de balanceo de carga todo el tráfico sala por una interfaz (`FastEthernet0/0` en nuestro caso) aunque haya dos cuyo coste al destino sea el mismo.

3. Reparto con routers reales

Tal y como se indica en el guión de la práctica ahora vamos a generar tráfico con `mgen` y estudiar las gráficas que obtenemos. ASumiremos que el equipo que genera tráfico es `SRC` y el que lo recibe será `DST`.

3.1. Prueba sin reparto de tráfico

En este caso veremos que todo el tráfico que enviemos desde `SRC` a `DST` seguirá la misma ruta a través del "rombo" en mitad de la topología. Como los enlaces de dicho rombo están limitados a

```
R1#sh interfaces stats
FastEthernet0/0
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        196      26261        415      41112
            Route cache         153      10710        165      17426
                  Total         349      36971        580      58538
FastEthernet0/1
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        200      26676        422      41573
            Route cache         177      15582        165      17434
                  Total         377      42258        587      59007
Ethernet1/0
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        164      17948        575      52270
            Route cache         330      34860        330      26292
                  Total         494      52808        905      78562
Interface Ethernet1/1 is disabled

Interface Ethernet1/2 is disabled

Interface Ethernet1/3 is disabled
```

Figura 6: Estadísticas de paquetes por interfaz antes de emitir tráfico de usuario.

```
R1#sh interfaces stats
FastEthernet0/0
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        246      32697        506      49835
            Route cache         156      10920        854      84948
                  Total         402      43617       1360     134783
FastEthernet0/1
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        254      33520        514      50637
            Route cache         877      84170        178      18780
                  Total        1131     117690        692      69417
Ethernet1/0
          Switching path    Pkts In   Chars In   Pkts Out  Chars Out
               Processor        172      18654        671      61284
            Route cache        1032     103728       1033      95090
                  Total        1204     122382       1704     156374
Interface Ethernet1/1 is disabled

Interface Ethernet1/2 is disabled

Interface Ethernet1/3 is disabled
```

Figura 7: Estadísticas de paquetes por interfaz tras emitir tráfico de usuario.

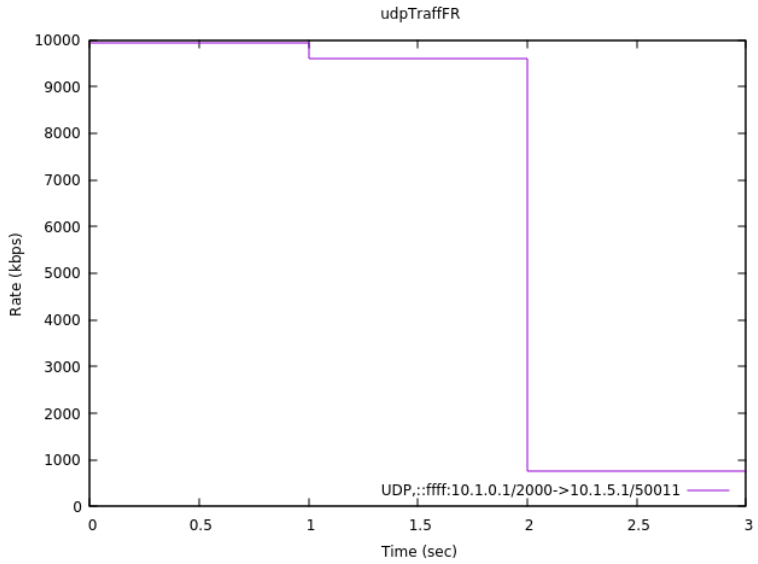

Figura 8: Tasa del flujo `UDP` sin reparto de tráfico.

10 $Mbps$ veremos cómo esta es la cota superior de la velocidad para ambos protocolos de transporte. En el caso de `UDP`, tal y como se ve en la figura 8, esta tasa de 10 $Mbps$ se mantendrá durante los 2 s que dura el flujo. Como la tasa del enlace por el que se desaloja el tráfico es menor que la tasa del flujo (22 $Mbps$) la cola de salida asociada a esta interfaz se irá llenando hasta provocar el descarte de paquetes. Tras finalizar el flujo se vaciará esta cola, cosa que explica la tasa no nula entre los segundos 2 y 3 donde el `SRC` ya no está emitiendo ningún tráfico. Podemos corroborar nuestras sospechas viendo que la latencia se satura rápidamente y que la tasa de pérdidas es prácticamente constante en todo momento ya que muchos de los paquetes emitidos por `SRC` se encuentran una cola llena.

En el caso de `TCP`, tal y como vemos en la figura 9, la tasa se comporta de manera parecida a la de `UDP`. La principal diferencia radica en que el intervalo de tiempo en el que mantenemos una tasa de 10 $Mbps$ es mayor ya que la propia operativa del protocolo "alarga" la ráfaga de tráfico para garantizar que llegue al receptor en su totalidad. Es por ello que observamos una tasa de pérdidas nula tal y como garantiza `TCP` por diseño.

En definitiva vemos que a grandes rasgos la tasa del flujo se limita a 10 $Mbps$ por los enlaces que conforman el rombo de la topología. Debido a este hecho, `UDP` perderá tráfico pero su ráfaga acabará "a tiempo" mientras que `TCP` transmite todo el tráfico, pero lo hace en un tiempo no predecible *a priori*.

3.2. Prueba con reparto de tráfico por paquete (PP)

Al activar el reparto de tráfico en ambos enlaces veremos cómo se incrementan las tasas que observamos en las gráficas de las figuras 10 y 11. Teóricamente deberíamos obtener gráficas escaladas por un factor próximo a 2 respecto a las de las figuras 8 y 9 ya que `R1` ahora hará uso de los dos "caminos" que componen el rombo del centro de la topología. El perfil del tráfico en sí es muy parecido al que vemos en la sección anterior pero las tasas son más altas al poder hacer uso de dos enlaces en vez de uno. No obstante, la tasa de la que hacemos uso es de $2 \cdot 10\ Mbps = 20\ Mbps < 22\ Mbps$ con lo que el flujo `UDP` sigue estando sujeto a pérdidas, aunque estas son claramente menores que en el caso anterior. Esto conlleva a que se entregue mucho más tráfico (el área de la gráfica es mayor) que en el escenario previo.

3.3. Prueba con reparto de tráfico por paquete y retardo

En este caso añadimos un retardo adicional de 5 ms a uno de los "caminos" del rombo. Veremos que esto hará que la latencia sea distinta en cada uno de los tramos y, dado que empleamos un reparto por paquete, se irá alternando con cada uno de ellos. En el caso de `UDP` esto se manifiesta en que se

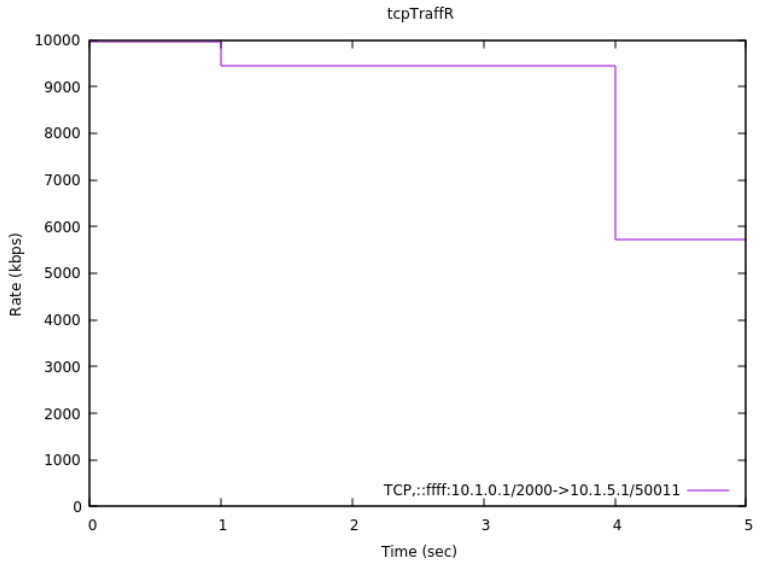

Figura 9: Tasa del flujo `TCP` sin reparto de tráfico.

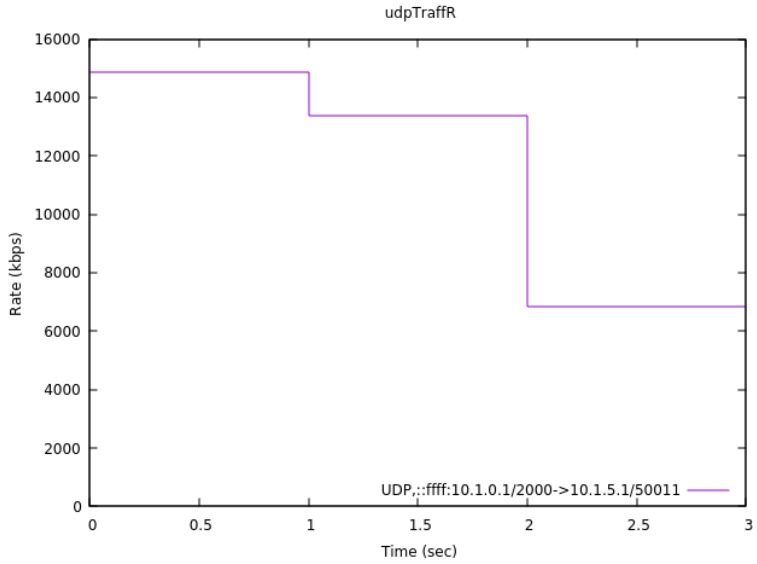

Figura 10: Tasa del flujo `UDP` con reparto de tráfico por paquete.

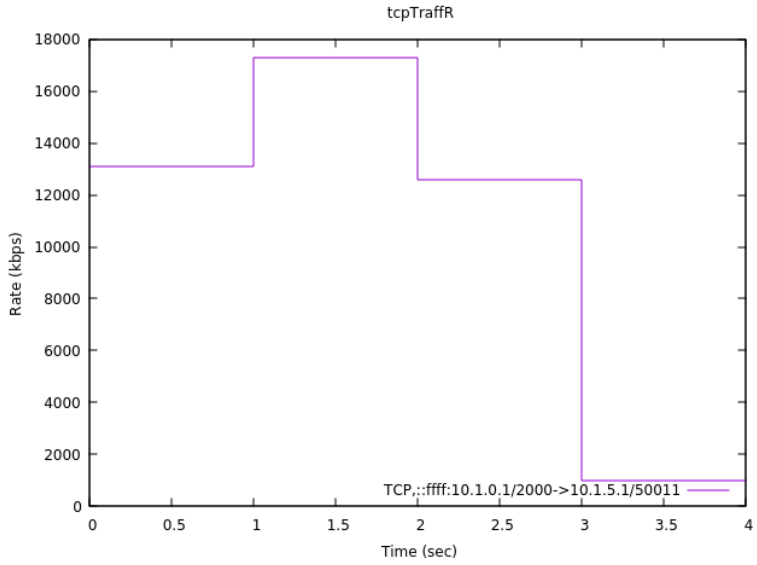

Figura 11: Tasa del flujo `TCP` con reparto de tráfico por paquete.

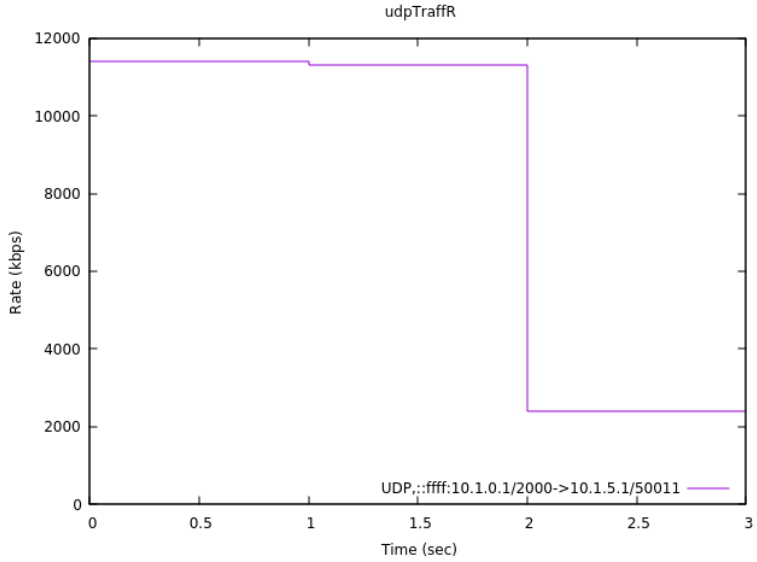

Figura 12: Tasa del flujo `UDP` con reparto de tráfico por paquete y latencia añadida.

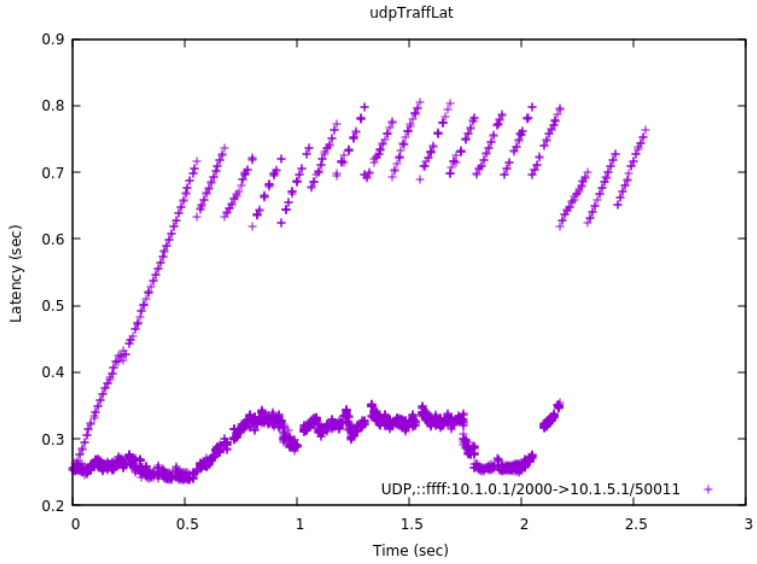

Figura 13: Latencia del flujo `UDP` con reparto de tráfico por paquete y latencia añadida.

rebaja la tasa de llegada en la figura 12 (uno de los caminos tarda 5 *ms* más en empezar a entregar datos) y en dos niveles de latencia diferentes en la figura 13. En el caso de `TCP` veremos que en la figura 14 el tiempo que se tarda en recibir todos los datos se incrementa considerablemente ya que cada uno de los paquetes que tome la ruta artificialmente lenta llegará fuera de secuencia, cosa que retrasará la entrega de los paquetes que lleguen por el camino rápido. De esta manera, el retardo de 5 *ms* se está aplicando a todo el flujo mientras que solo se aplica a la "mitad" en el caso de `UDP`.

3.4. Prueba con reparto por paquete y congestión

En este caso limitaremos la tasa de uno de los caminos a 5 *Mbps* artificialmente. En el caso de `UDP` veremos que la tasa que obtenemos es mayor que en el caso en el que no se aplica reparto de tráfico pero menor que cuando el sistema no estaba congestionado tal y como se aprecia en la figura 15. Esto se debe a que uno de los "caminos" tiene una menor tasa, lo que implica que en `R2` se formará otra nueva cola y que la tasa máxima a la que aspiremos sea de unos 15 *Mbps* sumando las de ambos "caminos". En el caso de `TCP` observaremos una tasa menor que la que teníamos en el escenario sin reparto de tráfico tal y como se aprecia en la figura 16. Ésto creemos que se debe a que para evitar la congestión que se produce en `R2` `TCP` controla la tasa de emisión a fin de no incrementar las pérdidas de manera desmesurada. Además, el hecho de que algunos paquetes se pierdan en `R2` provocará que

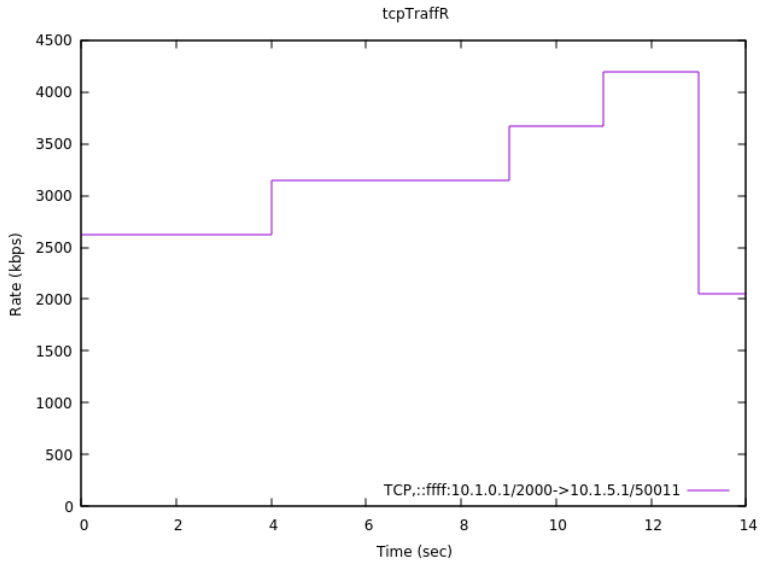

Figura 14: Tasa del flujo `TCP` con reparto de tráfico por paquete y latencia añadida.

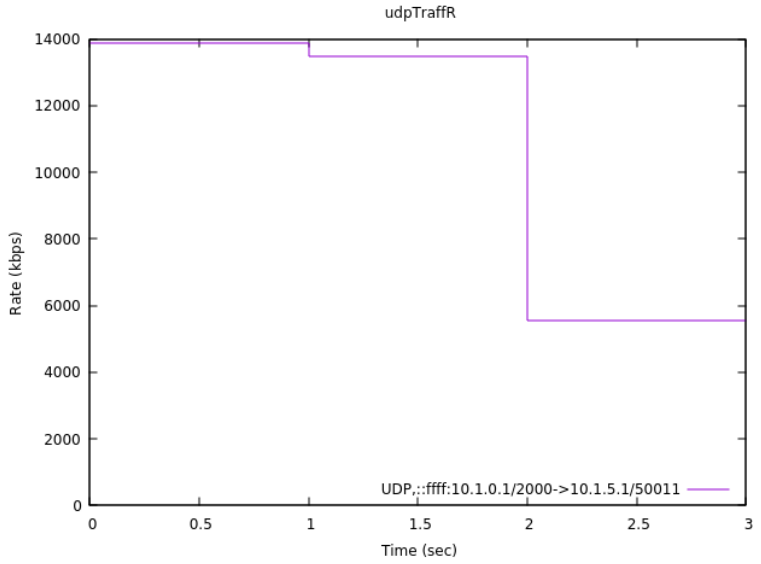

Figura 15: Tasa del flujo `UDP` con reparto de tráfico por paquete y congestión.

paquetes que alcancen `DST` deban esperar a la retransmisión de las pérdidas para ser entegados.

3.5. Tiempo de recuperación con reparto por destino (PD)

Para realizar esta prueba hemos empleado las `TTY`s del ofrecidas por el *Ubuntu Server* que ejecuta `SRC`. Para cambiar a la segunda consola tan solo hay que pulsar `ALT + F2` y podemos llevar a cabo la prueba sin problema. Además, hemos reconfigurado `R2` para que vuelva a limitar la velocidad a 10 *Mbps* como hacia originalmente.

Al aplicar el reparto de tráfico por destino veremos que cada `ping` coge una ruta distinta por el rombo ya que ambas tienen igual coste. Así, cuando se suspende un enlace provocaremos que una de las rutas quede inutilizada. No obstante, si observamos la tabla de encaminamiento de `R1` con `show ip route` veremos que la desaparición de una de las rutas tarda en propagarse un tiempo por la propia operativa de `OSPF`, el protocolo de encaminamiento que ha descubierto la ruta hasta la subred `10.1.5.0/24`. Por tanto, el tiempo de recuperación equivale al tiempo que tarda el proceso que implementa `OSPF` en `R1` en darse cuenta de que una de las rutas no está disponible. En este instante no tiene cabida el reparto de tráfico y ambos`ping`s irán por la misma ruta. Nosotros hemos observado que ésto toma unos 45 *s* para ocurrir en nuestro equipo y que este tiempo es prácticamente idéntico

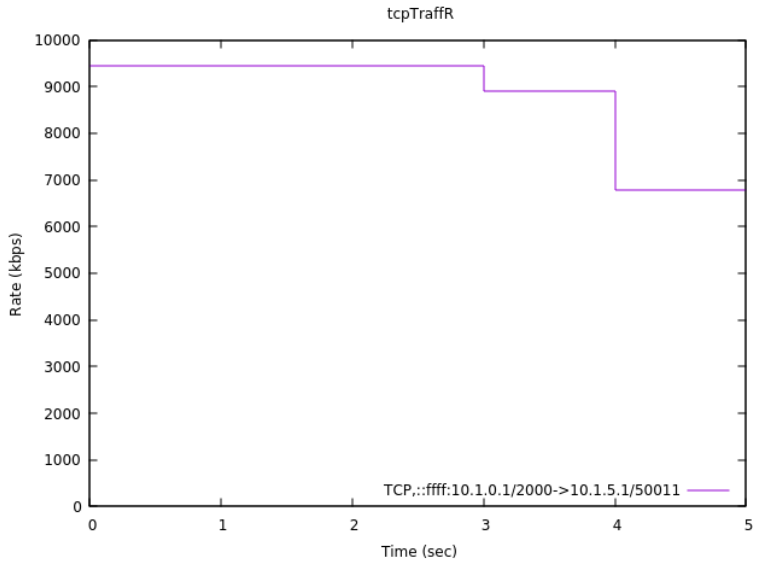

Figura 16: Tasa del flujo `TCP` con reparto de tráfico por paquete y congestión.

sin importar si suspendemos el enlace `R2-R4` o el `R1-R2`. Podemos señalar que al suspender el segundo, observamos un mensaje en la consola de `R1` que alude a que expira un *timer* asociado a dicho enlace, cosa que provoca la actualización de la tabla de encaminamiento. Si suspendemos el primer enlace el mensaje aludiendo al *timer* aparece en la consola de `R2` y acto seguido se reanuda el ping en cuestión. Como el tiempo de propagación de este cambio en `OSPF` es muy rápido, apenas notamos diferencias entre la suspensión de ambos enlaces.

Solo nos queda señalar que como la cadencia de los `ping`s es de 1 s, ya que no hemos configurado lo contrario, conocer el tiempo de recuperación equivale a restar el número de secuencia de `ICMP` (`icmp_seq`) del primer mensaje tras la recuperación y el último que se emitió al haber una correspondencia directa entre número de secuencia y segundo de emisión.

4. Conclusiones

Tras finalizar la práctica tenemos un entendimiento mucho más profundo del balanceo de carga en redes de computadores así como el efecto de esta posibilidad de escoger rutas, y las diferencias entre ellas, sobre flujos de tráfico que empleen los servicios tanto de `UDP` como de `TCP`.