Redes de Ordenadores para Robots y Máquinas Inteligentes
Práctica 4
Calidad de Servicio: Control de tráfico en Linux

GSyC
Departamento de Teoría de la Señal y Comunicaciones
y Sistemas Telemáticos y Computación
URJC

Marzo de 2024

Introducción

Descarga tu escenario de red para esta práctica del siguiente enlace:

`https://mobiquo.gsyc.urjc.es/practicas/ror/p4.html`

Descomprime el fichero que contiene el escenario de NetGUI `lab-tc.tgz`.

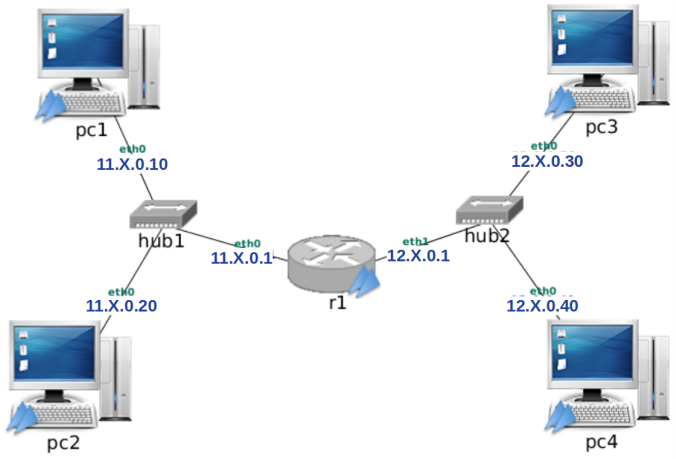

Figura 1: Escenario para control de tráfico

1. Control de Tráfico

1.1. Sin control de tráfico ni a la entrada ni a la salida

El *router* `r1` no tiene activado el control de tráfico en ninguna de sus interfaces.

1.1.1. Un flujo de datos

Inicia una captura en la interfaz `r1(eth1)` guardando el contenido en `tc-01.cap`.

Arranca `iperf` en modo servidor UDP en `pc3` y arranca `iperf` en modo cliente UDP en `pc1` para que envíe tráfico a 3M durante 10 segundos a `pc3`.

Observa en el servidor el informe que aparece al terminar de recibir el tráfico `pc1` → `pc3`.
Carga la captura en `wireshark` y muestra el flujo de forma gráfica, incluye una imagen en la memoria.

1.1.2. Dos flujos de datos

- Arranca `iperf` en modo servidor UDP en `pc4`.
- Arranca otro `iperf` en modo servidor UDP en `pc3`.
- Inicia una captura de tráfico en la interfaz `eth1` de `r1` (`tc-02.cap`).
- Escribe (todavía sin ejecutar) el comando que arranca `iperf` en modo cliente UDP en `pc1` para que envíe 3M al servidor `pc3` en el sentido `pc1` → `pc3` durante 10 segundos.
- Escribe (todavía sin ejecutar) el comando que arranca `iperf` en modo cliente UDP en `pc2` para que envíe 3M al servidor `pc4` en el sentido `pc2` → `pc4` durante 10 segundos.
- Ejecuta los dos comandos anteriores uno a continuación de otro (lo más rápidamente que puedas) para que su ejecución se realice de forma simultánea.
- Interrumpe la captura una vez que los clientes hayan terminado de ejecutar `iperf`.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Incluye una imagen en la memoria que muestre los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

1.2. Control de admisión para el tráfico de entrada

Vamos a configurar `r1` para restringir el tráfico de entrada distinguiendo 2 flujos de datos:

- Flujo 1: origen `pc1`, se quiere restringir con TBF a una velocidad de `1mbit`[1] y una cubeta de 10k bytes.
- Flujo 2: origen `pc2` se va a restringir a una velocidad de `2mbit` y una cubeta de 10k bytes.

Utiliza `tc` para definir esta configuración en la interfaz `eth0` de `r1`, que es la interfaz de entrada para los flujos 1 y 2. Haz que se aplique primero el filtro del flujo número 1 y después el del número 2. Guarda esta configuración en un fichero de *script* con el nombre `tc-ingress.sh`:

```
#!/bin/sh

# Esto es un comentario

echo "Borrando la disciplina de cola ingress en la interfaz eth0"
```

[1] En todo este enunciado, escribiremos las velocidades de transmisión con la notación que requiere la sintaxis de `tc`. Por lo tanto, escribiremos `1mbit` para referirnos a una velocidad de 1 Mbps

```
tc qdisc del ...

echo "Creando la disciplina de cola ingress en la interfaz eth0"
tc qdisc add ...
...
```

Una vez creado el *script* recuerda darle permisos de ejecución.

Si prefieres, puedes editar el script en la máquina real (Ubuntu) con un editor gráfico y luego ejecutarlo.

IMPORTANTE: Escribe el *script* de forma que sólo borre la disciplina de cola si está definida, para que no dé un error. Para ello, utiliza el comando `tc qdisc show dev eth0` y comprueba su salida. Si no devuelve nada, es que no hay ninguna qdisc definida, si devuelve algo es que hay una definida y conviene borrarla primero antes de añadirla.

Incluye dentro de la memoria el contenido del *script*, así como de los *scripts* que desarrolles en los siguientes apartados.

1. Consulta la configuración actual de las disciplinas de cola configuradas por defecto en `r1`. Indica qué resultado has obtenido para cada una de las colas.

2. Realiza una prueba de tráfico como la del apartado anterior:

   - Inicia una captura de tráfico en la interfaz `eth1` de `r1` (`tc-03.cap`)
   - Arranca dos clientes y 2 servidores tal y como lo hiciste en el apartado 1.1.2.
   - Interrumpe las capturas cuando los servidores hayan terminado de recibir todo el tráfico.

3. Explica las estadísticas que muestran los servidores.

4. Carga las capturas en `wireshark` y muestra cada uno de los flujos de forma gráfica. Incluye una imagen en la memoria que muestre los flujos de ambas capturas de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

5. Consulta la configuración actual de las disciplina de cola configurada a la entrada en `eth0`. Indica el número de paquetes recibidos y el número de paquetes descartados.

1.3. Disciplinas de colas para el tráfico de salida

1.3.1. Token Bucket Filter (TBF)

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh`.

- Define en `r1` para su interfaz `eth1` una disciplina TBF de salida con tasa de envío de `1.5mbit`, tamaño de cubeta 10k y latencia 10 ms, y guarda la configuración en un nuevo *script* `tc-egress-tbf.sh`.

- Inicia una captura de tráfico en la interfaz `eth1` de `r1` (`tc-04.cap`).

- Arranca 2 clientes y 2 servidores tal y como lo hiciste en el apartado 1.1.2.

- Interrumpe la captura cuando los servidores hayan terminado de recibir todo el tráfico.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.

2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Modifica la configuración de TBF de salida para que ahora tenga una latencia de 20 segundos (NOTA: ahora son 20 segundos en vez de 10 milisegundos) y realiza la misma prueba que antes [2].

Llama ahora a la captura `tc-05.cap`.

Interrumpe la captura cuando haya pasado tiempo suficiente para que termine de llegar todo el tráfico a los servidores (será unos 20 segundos después de que comenzó a enviarse). A continuación analiza los resultados obtenidos:

3. Explica las estadísticas que muestran los servidores.

4. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Incluye una imagen en la memoria que muestre los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

5. ¿Cuánto tiempo ha tardado `r1` en realizar el reenvío de todo el tráfico? Relaciona este valor con la cantidad de datos que tenía que reenviar y la tasa de envío que estaba utilizando `r1`. Del tráfico originalmente enviado por `pc1` y `pc2`, ¿cuánto ha sido descartado en la disciplina de cola asociada a `eth0` de `r1`? ¿Y cuánto ha sido descartado en la disciplina asociada a `eth1` de `r1`?

1.3.2. TBF + PRIO

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh`. Borra la disciplina de cola de salida TBF configurada en la interfaz `eth1` de `r1`.

La configuración TBF en el apartado 1.3.1 permite gestionar la tasa de envío para que no supere el valor configurado, en nuestro caso 1.5Mbit. Esta disciplina de cola es sin clases y trata a todos los paquetes por igual. Ahora vamos a querer fijar la tasa de envío de `r1` pero tratando los paquetes con diferentes prioridades.

Toma como punto de partida esta configuración para que ahora se atienda el tráfico de salida según diferentes prioridades, configurando una disciplina de cola con prioridad que sea hija de la disciplina TBF.

- Escribe un *script* en `r1`, `tc-egress-tbf-prio.sh`, para configurar TBF con los siguientes parámetros: ancho de banda `1.5mbit`, cubeta 10k y latencia 20s. Crea una disciplina de cola hija con prioridad de tal forma que se asignen las siguientes prioridades:
  - Prioridad 1 (más prioritario): tráfico de la dirección IP origen `pc1`
  - Prioridad 2 (prioridad intermedia): tráfico de la dirección IP origen `pc2`
  - Prioridad 3 (menos prioritario): sin definir, pues no lo necesitamos.

[2] Ten en cuenta que ahora el tráfico quedará en la cola de la disciplina TBF esperando a ser cursado según la tasa de envío que hemos configurado. El cliente terminará de enviar a los 10 segundos y esperará a recibir el informe del servidor. Sin embargo, el servidor no acabará de recibir (y por tanto no enviará el informe) hasta que TBF no termine de atender el tráfico de la cola de salida, que será más de 10 segundos. Al no recibir el cliente el informe del servidor, terminará imprimiendo un `Warning`. De la misma forma cuando el servidor haya terminado de recibir y envíe el informe al cliente, éste ya habrá terminado su ejecución e imprimirá un mensaje indicando que no ha podido enviar el informe al cliente: `Connection refused`.

- Inicia una captura de tráfico en la interfaz `eth1` de `r1` (`tc-06.cap`).
- Arranca 2 clientes y 2 servidores tal y como lo hiciste en el apartado 1.1.2.
- Interrumpe la captura cuando haya pasado tiempo suficiente para que termine de llegar todo el tráfico a los servidores (será unos 20 segundos después de que comenzó a enviarse).

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en wireshark y muestra cada uno de los flujos de forma gráfica. Incluye una imagen en la memoria que muestre los flujos de forma gráfica. Explica la evolución en el tiempo del ancho de banda medido para cada uno de los flujos.

1.3.3. Hierarchical token Bucket (HTB)

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh`. Borra la disciplina de cola de salida configurada en la interfaz `eth1` de `r1`.

- Escribe un *script* en `r1`, `tc-egress-htb.sh`, para configurar en su interfaz `eth1` una disciplina HTB de salida con ancho de banda `1.2mbit`. Reparte el ancho de banda de esta interfaz de salida de la siguiente forma:
  - `700kbit` para el tráfico con origen en `pc1`, `ceil 700kbit`.
  - `500kbit` para el tráfico con origen en `pc2`, `ceil 500kbit`.
- Inicia una captura de tráfico en la interfaz `eth1` de `r1` (`tc-07.cap`).
- Arranca 2 clientes y 2 servidores tal y como lo hiciste en el apartado 1.1.2.
- Interrumpe la captura cuando haya pasado tiempo suficiente para que termine de llegar todo el tráfico a los servidores (será unos 20 segundos después de que comenzó a enviarse). `iperf`.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica la evolución en el tiempo del ancho de banda medido para cada uno de los flujos.

Modifica la configuración de `ceil` en cada uno de los flujos para que puedan utilizar `1.2Mbit`. Realiza la misma prueba que antes capturando de nuevo el tráfico (`tc-08.cap`) y analiza los resultados obtenidos:

3. Explica las estadísticas que muestran los servidores.
4. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Incluye una imagen en la memoria que muestre los flujos de forma gráfica. Explica la evolución en el tiempo del ancho de banda medido para cada uno de los flujos.

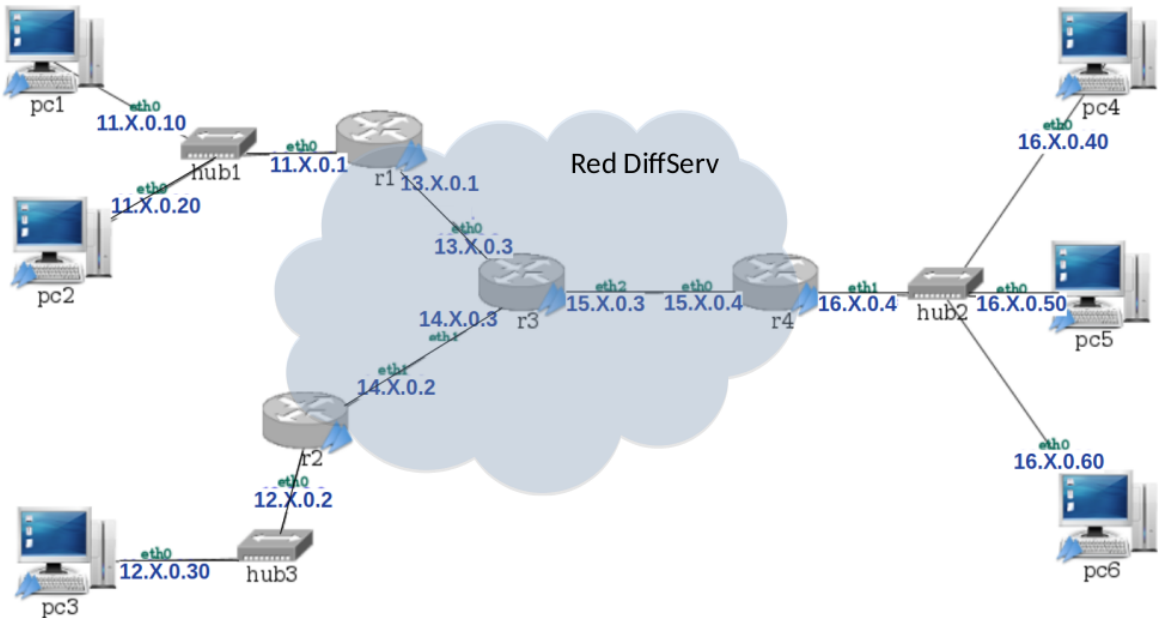

Figura 2: Escenario para DiffServ

2. Diffserv

Descarga tu escenario de red para esta práctica del siguiente enlace:

`https://mobiquo.gsyc.urjc.es/practicas/ror/p4.html`

Descomprime el fichero que contiene el escenario de NetGUI lab-DiffServ.tgz para realizar la práctica de diffServ en Linux.

En el escenario de la figura 2 se va a configurar la red para que el tráfico desde `pc1`, `pc2` y `pc3` envíen paquetes a `pc4`, `pc5` y `pc6` atravesando una red diffServ. Configura las direcciones IP en tu escenario utilizando las tus 4 subredes de la práctica 1, y elige las subredes que quieras como subdred 13.X.0.0/24 y subdred 14.X.0.0/24.

Para esta práctica se distinguirán 4 calidades diferentes, con los códigos EF, AF31, AF21 y AF11.

2.1. Configuración de función policing y marcado de tráfico en DSCP

Utiliza la herramienta `tc` para garantizar que el tráfico que entra en `r1` cumple las siguientes características:

- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc1`, descartando el tráfico sobrante:
  - Flujo 1: máximo 1.2mbit con ráfaga 10k para el tráfico dirigido a `pc4`, marcado con calidad EF. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 2.
  - Flujo 2: máximo de 600kbit y ráfaga 10k, marcado con calidad AF31. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r1`.
- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc2`, descartando el tráfico sobrante:
  - Flujo 3: máximo 300kbit con ráfaga 10k para el tráfico dirigido a `pc5`, marcado con AF21. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 4.

- Flujo 4: máximo de 400kbit y ráfaga 10k, marcado con AF11. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r1`.

Utiliza la herramienta `tc` para garantizar que el tráfico que entra en `r2` cumple las siguientes características:

- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc3`, descartando el tráfico sobrante:
  - Flujo 5: máximo 400kbit con ráfaga 10k dirigido a `pc6`, marcado con AF31. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 6.
  - Flujo 6: máximo 300kbit con ráfaga 10k dirigido a `pc6`, marcado con AF21. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 7.
  - Flujo 7: máximo 100kbit con ráfaga 10k, marcado con AF11. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r2`.

1. Realiza *scripts* para `r1` y otro para `r2` donde se configuren estos perfiles de tráfico. Incluye dichos *scripts* en la memoria.

2. Inicia capturas: `diffServ-01.cap` en la subdred 13.X.0.0/24, `diffServ-02.cap` en la subdred 14.X.0.0/24 y `diffServ-03.cap` en la subdred 15.X.0.0/24 para que capture el tráfico que se genera en tu escenario por el envío "simultáneo" de:
   - Desde el `pc1` 2M a `pc4`
   - Desde el `pc2` 1.5M a `pc5`
   - Desde el `pc3` 1M a `pc6`

3. Interrumpe las capturas, al menos 1 minuto después de que la transmisión haya terminado. Comprueba que el resultado es el esperado:
   - El tráfico que entra en la red diffServ es el que se ha especificado en el control de admisión.
   - El tráfico está marcado según las especificaciones anteriores.

   Para ello, consulta las gráficas `IO graphs` de Wireshark aplicando los filtros sobre las marcas DSCP de tal forma que se muestre cada calidad marcada de cada una de las fuentes:
   - Tráfico de EF
   - Tráfico de AF31
     - Total
     - Con origen en `pc1`.
     - Con origen en `pc3`.
   - Tráfico de AF21
     - Total
     - Con origen en `pc2`.
     - Con origen en `pc3`.
   - Tráfico de AF11
     - Total
     - Con origen en `pc2`.
     - Con origen en `pc3`.

   Explica los resultados obtenidos e incluye todas las gráficas que consideres necesarias en la memoria.

2.2. Tratamiento de tráfico en función del marcado DSCP

Mantén la configuración realizada en `r1`, `r2`.

Se establecen los siguientes parámetros de calidad dentro del router del núcleo diffServ (`r3`) para cada una de las calidades definidas. Configura HTB con ancho de banda 2.4Mbit para compartir entre todos los flujos con el siguiente patrón:

- EF: HTB 1Mbit como mínimo y 1Mbit como máximo.
- AF31: HTB 500kbit como mínimo y 500kbit como máximo.
- AF21: HTB 400kbit como minimo y 400kbit como máximo.
- AF11: HTB 200kbit como mínimo y 200kbit como máximo.

1. Realiza un *script* para `r3` donde se configure esta disciplina de cola según el marcado de los paquetes e incluye dicho *script* en la memoria.
2. Inicia una captura (`diffServ-04.cap`) en la subdred 15.X.0.0/24 para que capture el tráfico que se genera en tu escenario por el envío "simultáneo" de:
   - Desde `pc1`: 2M a `pc4`
   - Desde `pc2`: 1.5M a `pc5`
   - Desde `pc3`: 1M a `pc6`

   Espera al menos 2 minutos después de que haya terminado de enviarse el tráfico de `pc1`, `pc2` y `pc3` antes de interrumpir la captura de tráfico.
3. Comprueba que el resultado es el esperado, es decir, el tráfico sigue el perfil indicado en las especificaciones anteriores. Para ello, consulta las gráficas `IO graphs` de Wireshark aplicando los filtros sobre las marcas DSCP de tal forma que se muestre cada calidad marcada de cada una de las fuentes incluyendo dichas imágenes en la memoria:
   - Tráfico de EF
   - Tráfico de AF31
   - Tráfico de AF21
   - Tráfico de AF11

   Explica los resultados obtenidos y explica si alguno de los flujos ha encolado tráfico para enviarlo posteriormente a los 10 segundos que dura la transmisión de `iperf`.
4. Modifica la configuración de HTB en `r3` para que si algún flujo no está utilizando el ancho de banda que tiene garantizado lo puedan usar el resto de flujos y vuelve a hacer una captura de tráfico (`diffServ-05.cap`) en la subdred 15.X.0.0/24. Explica qué modificaciones has tenido que hacer en el *script*.
5. Explica los resultados obtenidos e incluye las gráficas `IO graphs` que consideres necesarias.

Normas de entrega

Es necesario entregar a través del aula virtual los siguientes ficheros:

- Memoria en formato pdf donde se explique razonadamente cada uno de los apartados de este enunciado y se incluya el contenido de los scripts que hayas desarrollado.
- Fichero `p4.tgz` o `p4.zip` resultado de comprimir **una carpeta de nombre p4** que contenga en su interior los ficheros de captura de tráfico: `tc-01.cap` hasta `tc-08.cap` y desde `diffServ-01.cap` hasta `diffServ-06.cap`.