Escuela Superior de Informática

Universidad de Castilla-La Mancha
Escuela Superior de Informática

Planificación e Integración de Sistemas y Servicios.

4º Grado en Ingeniería Informática.

Práctica 4. Gestión de prioridad de tráfico en IP.

Autor:
Alberto Salas Seguín y
Marcos López Sobrino.

Fecha:
1 de enero de 2019

1. Entorno de trabajo.

El entorno de trabajo utilizado han sido 3 máquinas virtuales gestionadas mediante Vagrant, con la imagen ***ubuntu/trusty64***. Una de ellas actúa como cliente, otra como servidor y otra como router entre ambas. A continuación, explicamos la configuración de las redes, de las máquinas, y los paquetes instalados.

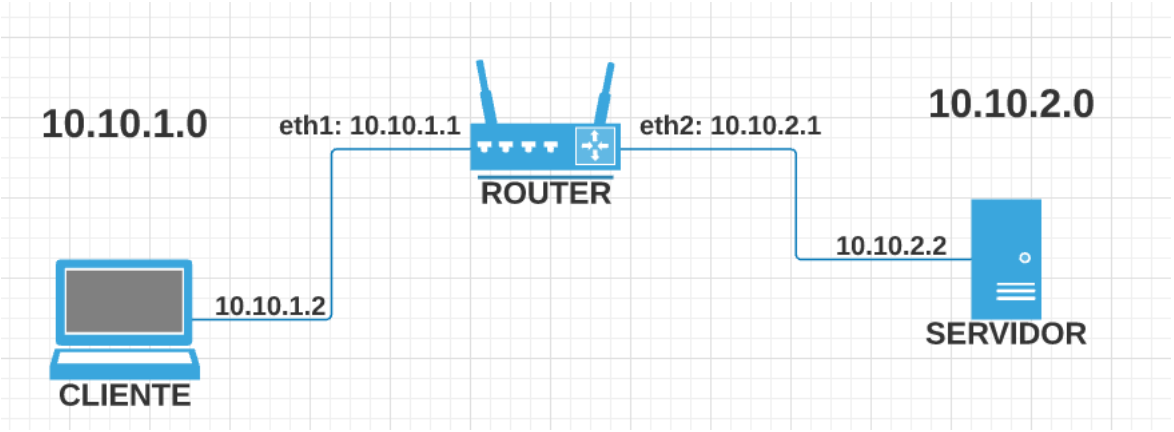

Figura 1: Topología de la red.

1.1. Cliente y servidor.

Como vemos en la figura anterior, la red del cliente, que corresponde a la interfaz **eth1** del router, tiene como IP **10.10.1.0**; mientras que la red del servidor, que corresponde a la interfaz **eth2** del router, tiene como IP **10.10.2.0**.

Para el correcto funcionamiento de las comunicaciones, hay que indicarle a la máquina cuál es la ruta por defecto, para ello, se ejecutan los siguientes comandos:

```
$ sudo ip route del default dev eth0
$ sudo ip route add default via <ip_router> dev eth1
```

donde *ip_router* sería la dirección IP de la interfaz del router dentro de la red correspondiente.

Los paquetes necesarios para el funcionamiento de la práctica en el caso del cliente y el servidor son:

- **iptables**
- **iperf**
- **sip-tester**

1.2. Router.

A la máquina del router se le ha instalado un entorno gráfico para poder usar cómodamente la herramienta **Wireshark**. Además, se le han asignado 2048 MB de RAM. Por otra parte, se ha configurado las interfaces de red con la dirección **.1** de la red correspondiente, tal y como vemos en la figura. Adicionalmente, para que la máquina actúe como router se debe activar el **IP forwarding**, para ello, en el playbook correspondiente del router se ha añadido una regla específica.

Por último, en el router se han instalado los siguientes paquetes:

- Para la interfaz gráfica:
  - **xorg**
  - **gnome-core**
  - **gnome-system-tools**
  - **gnome-app-install**
- **wireshark**
- **iptables**

1.3. Archivos de configuración e inicialización.

Todo lo anterior se recoge en el archivo ***Vagrantfile*** y los distintos ***playbooks***, que se pueden encontrar junto a este documento. Para poner en funcionamiento el entorno de trabajo, ejecutamos:

```
vagrant up --provision
```

2. Marcado de paquetes.

2.1. Configuración de iptables en router, cliente y server.

Dado que tanto en la máquina del router como la del cliente, se van a introducir reglas de iptables para el correcto funcionamiento, para no perder estas reglas se va a utilizar ***iptables-persistent***, el modo para recuperar todas las reglas cada vez que se inicie la máquina es mediante una *task* cuya función es cargar un archivo con todas las reglas necesarias en el directorio ***/etc/iptables/rules.v4***, este directorio se encuentra tanto en la máquina del router como en la máquina del cliente. La regla que hemos utilizado para copiar este archivo en el directorio mencionado es la siguiente:

```
- name: copy iptables
  copy:
    src: <archivo a copiar>
    dest: /etc/iptables/rules.v4
    owner: root
    group: root
    mode: 0644
  notify: "restore iptables"
```

Como vemos en el código anterior, en la última línea se hace mención a ***notify***, lo cual hace necesario la presencia de un manejador, de manera que cada vez que se invoque a la tarea ***copy tables***, se va a hacer el *notify*, cuya función es hacer un **restore** del archivo **/etc/iptables/rules.v4** que hemos copiado. Esto lo podemos ver en el siguiente código:

```
handlers:
    - name: restore iptables
      shell: iptables-restore /etc/iptables/rules.v4
      listen: "restore iptables
```

2.2. Reglas iptables utilizadas en router, cliente y server.

- **Router.**

  En el caso del router hemos utilizado las reglas iptables que vienen dadas en el enunciado de la práctica. Estas son:

```
iptables -t filter -A INPUT -m dscp --dscp 14 -m limit --
   limit 5/s --limit-burst 5 -j ACCEPT

iptables -t filter -A INPUT -m dscp --dscp 14 -j DROP
```

  Donde los parámetros –limit y –limit-burst significan lo siguiente.

  - **limit.** El módulo limit se utiliza para restringir la tasa de coincidencias, por ejemplo, para suprimir los mensajes de registro. Sólo coincidirá con un número determinado de veces por segundo (por defecto 3 coincidencias por hora, con una ráfaga de 5). Se necesitan dos argumentos opcionales:
    - **-limit.** Seguido de un número y una unidad, especifica el número medio máximo de coincidencias que puede permitirse por segundo. Nos permite configurar el tiempo que tiene que pasar para que esta regla se cumpla entre dos paquetes.
    - –**limit-burst.** Seguido de un número que indica la cantidad máxima de paquetes antes de que se active el límite anterior.

- **Cliente y servidor.** Dado que vamos a marcar paquetes, la tabla que vamos a usar es **mangle**. Por un lado, el tráfico RTP lo vamos a marcar con **Expedited Forwarding**, esta clase es la máxima prioridad para el marcado de paquetes RTP, debido a que no queremos una pérdida de paquetes, ya que estos paquetes contienen la voz de una comunicación, por lo que en caso de pérdida, la conversación no se desarrollaría de forma completa. Por otro lado, el tráfico SIP lo vamos a marcar como **Assured Forwarding 11**, este tráfico se utiliza para la señalización durante la llamada y establecer las comunicaciones, por lo que no es necesario marcarlo con una máxima prioridad como es el caso del tráfico RTP. Por último, el resto de paquetes, los vamos a marcar como **Best Effort**, es la categoría con menor prioridad en el marcado de paquetes.

Las reglas nos quedan de la siguiente manera:

```
/---- Trafico RTP ----/
iptables -t mangle -A OUTPUT -p udp -m udp --dport 10000:20000
   -j DSCP --set-dscp-class ef

/---- Trafico SIP ----/
iptables -t mangle -A OUTPUT -p udp -m udp --dport 5060 -j DSCP
    --set-dscp-class af11

/---- Resto de paquetes ----/
iptables -A OUTPUT -j DSCP --set-dscp 0
```

- La primera regla permite el tráfico udp saliente que tiene como destino el rango de puertos que corresponden al protocolo RTP (10000:20000) y dicho tráfico se ha marcado como Expedited Forwarding.
- La segunda regla permite el tráfico udp saliente que tiene como puerto destino (dport) el puerto 5060, el cual es el puerto dedicado al protocolo SIP. Como hemos dicho antes, este tráfico se ha marcado con AF11.
- La tercera regla se utiliza para el marcado de paquetes Best Effort.

3. Análisis del rendimiento.

Para la realización de este apartado se va a utilizar la herramienta **iperf**. Lo primero que vamos a hacer es utilizar la herramienta para ver cual es el ancho de banda sin incluir las reglas iptables definidas en el apartado anterior. Para establecer la comunicación:

```
$ iperf -s                          // Máquina del server.
$ iperf -c 10.10.2.2 -i 2           // Máquina del router.
```

Tras realizar esta prueba, los resultados son los siguientes:

```
root@router:/home/vagrant# iperf -c 10.10.2.2 -i 2
------------------------------------------------------------
Client connecting to 10.10.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.2.1 port 37999 connected with 10.10.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   290 MBytes  1.22 Gbits/sec
[  3]  2.0- 4.0 sec   301 MBytes  1.26 Gbits/sec
[  3]  4.0- 6.0 sec   301 MBytes  1.26 Gbits/sec
[  3]  6.0- 8.0 sec   293 MBytes  1.23 Gbits/sec
[  3]  8.0-10.0 sec   293 MBytes  1.23 Gbits/sec
[  3]  0.0-10.0 sec  1.44 GBytes  1.24 Gbits/sec
```

Si ahora esta misma prueba la hacemos incluyendo las llamadas de **sipp**, el resultado es el siguiente:

```
root@router:/home/vagrant# iperf -c 10.10.2.2 -i 2
------------------------------------------------------------
Client connecting to 10.10.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.2.1 port 38063 connected with 10.10.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   230 MBytes   964 Mbits/sec
[  3]  2.0- 4.0 sec   251 MBytes  1.05 Gbits/sec
[  3]  4.0- 6.0 sec   234 MBytes   983 Mbits/sec
[  3]  6.0- 8.0 sec   241 MBytes  1.01 Gbits/sec
[  3]  8.0-10.0 sec   238 MBytes   999 Mbits/sec
[  3]  0.0-10.0 sec  1.17 GBytes  1.00 Gbits/sec
```

Vemos que ya el ancho de banda es más bajo que cuando únicamente se utilizaba iperf. Esta reducción es debido al tráfico generado por *sipp*.

Llegados aquí, introducimos las reglas iptables para realizar las mismas pruebas anteriores.
Con las reglas iptables activadas y con iperf, el resultado es el siguiente:

```
Client connecting to 10.10.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.2.1 port 53610 connected with 10.10.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   269 MBytes  1.13 Gbits/sec
[  3]  2.0- 4.0 sec   276 MBytes  1.16 Gbits/sec
[  3]  4.0- 6.0 sec   285 MBytes  1.20 Gbits/sec
[  3]  6.0- 8.0 sec   290 MBytes  1.22 Gbits/sec
[  3]  8.0-10.0 sec   296 MBytes  1.24 Gbits/sec
[  3]  0.0-10.0 sec  1.38 GBytes  1.19 Gbits/sec
```

Observamos de nuevo, que el ancho de banda es menor que en el primer listado de código, donde no se utilizaba ni *iptables* ni *sipp*. Con esto deducimos que el marcado de los paquetes está funcionando bien.

La última prueba es utilizando las reglas *iptables* y *sipp*, tras ejecutar dicha prueba el resultado es el siguiente:

```
root@router:/home/vagrant# iperf -c 10.10.2.2 -i 2
------------------------------------------------------------
Client connecting to 10.10.2.2, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 10.10.2.1 port 53698 connected with 10.10.2.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   239 MBytes  1.00 Gbits/sec
[  3]  2.0- 4.0 sec   241 MBytes  1.01 Gbits/sec
[  3]  4.0- 6.0 sec   251 MBytes  1.05 Gbits/sec
[  3]  6.0- 8.0 sec   262 MBytes  1.10 Gbits/sec
[  3]  8.0-10.0 sec   250 MBytes  1.05 Gbits/sec
[  3]  0.0-10.0 sec  1.21 GBytes  1.04 Gbits/sec
```

De nuevo, se ha reducido el ancho de banda. Se están produciendo descartes de paquetes marcados como Best Effort, mientras que los paquetes RTP y SIP tienen el ancho de banda necesario para sus comunicaciones.

4. Bibliografía

- Configuración reglas Iptables.

  https://netfilter.org/documentation/HOWTO/packet-filtering-HOWTO-7.html#ss7.3

  https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/10103-dscpvalues.html

  https://community.rti.com/kb/how-set-dscp-flag-your-system

- Acerca de sipp.

  http://sipp.sourceforge.net/doc/reference.html