Redes Multimedia

Práctica 4: Ingeniería de tráfico

Contenido

1 Introducción

En las redes multimedia, es habitual que los elementos de conmutación implementen algoritmos que les permitan gestionar de forma adecuada el tráfico de los distintos servicios que se distribuyen en dichas redes. De esta manera, se pueden garantizar ciertos valores de calidad de servicio (tasa de transferencia, retardo, etc.)

Por ejemplo, se suele hablar típicamente de mecanismos de conformado de tráfico (*traffic shaping*) o de políticas de tráfico (*traffic policing*). En el caso de *traffic shaping*, se regula la tasa a la que se transmite cierto flujo de tráfico, de forma que no rebase en promedio un valor concreto en bits por segundo, retrasando el envío de aquellos paquetes que excedan dicha tasa (y por tanto, añadiendo latencia a dichos paquetes). La Figura 1 muestra cómo varía el tráfico al realizar un conformado de tráfico:

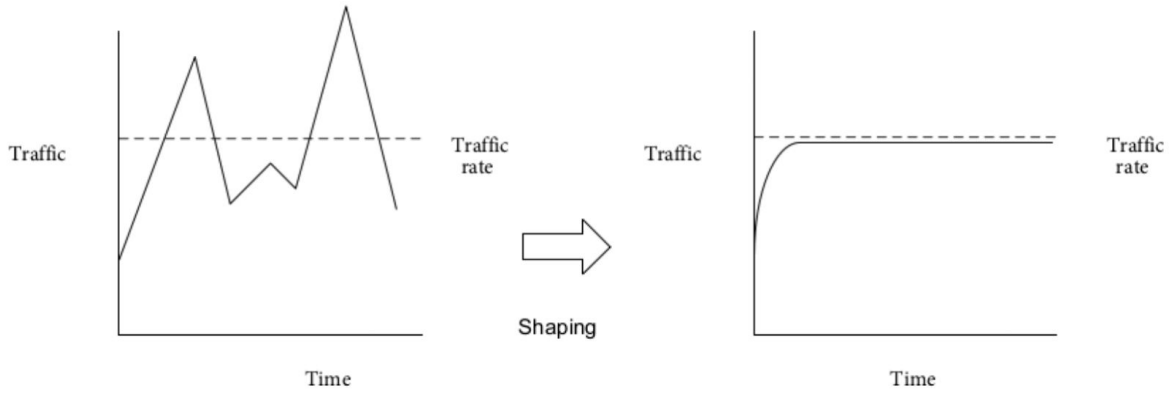

Figura 1 - Aplicación de conformado de tráfico: los picos de tráfico desaparecen, y se encolan, rellenándose con ellos los valles.

En el caso de *traffic policing* se regula igualmente la tasa de transmisión de cada tipo de flujo, pero en este caso aquellos paquetes que excedan la tasa serán marcados como de baja prioridad, o directamente descartados. Los paquetes no descartados, a diferencia del caso previo, no verán aumentar su latencia. La Figura 2 ilustra este segundo caso:

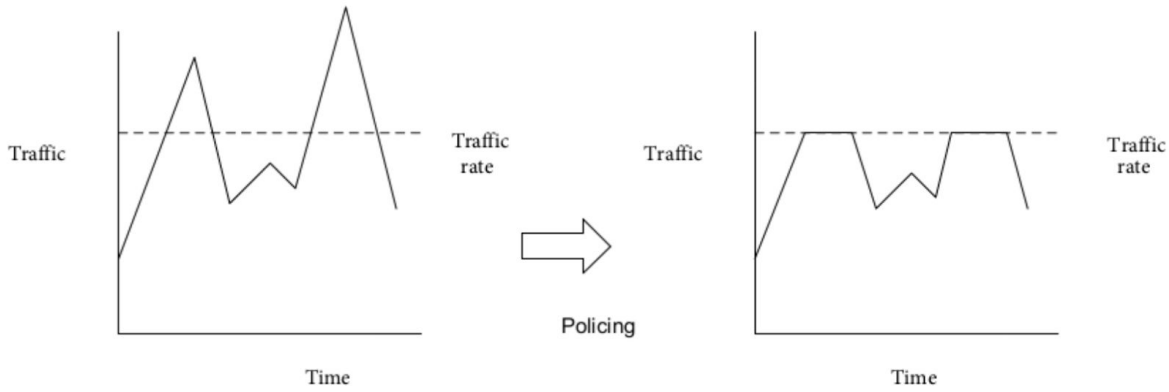

Figura 2 - Aplicación de política de tráfico: los picos de tráficos desaparecen, al descartarse ese tráfico. Los valles siguen existiendo.

Para implementar estos mecanismos de asignación de capacidad se pueden emplear disciplinas de colas tales como WFQ (*Weighted Fair Queuing*, cola equitativa ponderada), SRR (*Shaped Round Robin*); o distintos algoritmos, como el cubo con goteo (*leaky bucket*) o el cubo con fichas (*token bucket*). En esta práctica nos centraremos en este último algoritmo, y veremos su utilidad para hacer conformado de tráfico.

2 Objetivo de la práctica

El objetivo fundamental de esta práctica es introducir al alumno en los mecanismos existentes en las redes multimedia para proporcionar mecanismos de control de la calidad proporcionada. Para ello, se pretende alcanzar los siguientes subobjetivos:

1. Estudiar la técnica del cubo de fichas o *token bucket*.
2. Estudiar cómo afecta dicha técnica a flujos multimedia.
3. Comparar la técnica de conformado de tráfico con la de política de tráfico.
4. Aplicar los conocimientos aprendidos en prácticas y asignaturas anteriores.

3 Cubo con fichas (*token bucket*)

El cubo con fichas sigue el algoritmo ilustrado en la Figura 3:

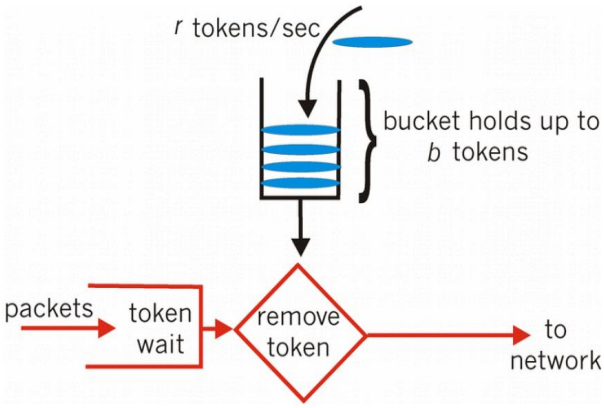

Figura 3 - Algoritmo de token-bucket

El algoritmo tiene un cubo en el que caben hasta ***b*** fichas o *tokens*, que podrá entenderse como el tamaño máximo que podrá tener una ráfaga o MBS (*Maximum Burst Size*). Dicho cubo se va llenando con una tasa de ***r*** tokens por segundo, que será equivalente a la tasa comprometida de transmisión o CIR (*Commited Information Rate*). Cuando el cubo se llena, ya no se sigue llenando hasta que haya espacio para volver a rellenar.

Los paquetes que llegan se almacenan en una cola a la espera de que haya tokens disponibles para enviar ese paquete. Cuando hay *tokens* suficientes, se quitan los tokens y se envía el paquete.

En la práctica haremos que cada byte del paquete (incluyendo todas sus cabeceras, CRC, preámbulo, hueco entre tramas) valga un *token*. A su vez, iremos llenando el cubo cada cierto tiempo, con tantos tokens como un paquete de tamaño máximo (incluyendo nuevamente todas sus cabeceras). Esto permite que mandar paquetes pequeños cueste menos *tokens*, y mandar paquetes grandes más tokens. De esta manera, se consigue controlar la tasa de transmisión en bits por segundo.

3 Realización de la práctica

1. (0,5 puntos) Estudie el código proporcionado, `emuladorTB.py`. ¿Cuáles son los parámetros del algoritmo (*b* y *r*) que se manejan para los distintos escenarios propuestos en el código? Indique el CIR en bits/s y MBS en bytes que se deberían alcanzar para el escenario que coincide con su número de pareja.
2. Utilice el `clienteTren2.py` que se facilita para esta práctica (equivalente al clienteTren2 solicitado en la Práctica 2) que puede limitar la tasa de transmisión en Kbit/s. Ejecute el `emuladorTB.py` utilizando el número de escenario que coincide con su número de pareja, transmita un tren de 1000 paquetes de 1460 bytes de datos de RTP, limitando la tasa al CIR calculado previamente y 10.000 Kbit/s.
   2.1. (3 puntos) Capture el tráfico del emulador con Wireshark, cara a realizar medidas de ancho de banda y retardo, como ya realizó en la Práctica 2. Represente con la función IOgraph de Wireshark el caudal **en bits/s** consumido por los 1000 paquetes a lo largo del tiempo, tanto a la entrada como a la salida del emulador (ambos flujos se pueden diferenciar fácilmente según su puerto de destino, excluyendo el tráfico ICMP). Explique los resultados obtenidos.
   2.2. (2 puntos) Mida a partir de los datos capturados a la salida del emulador el retardo de los paquetes (entendido como la diferencia entre el tiempo de transmisión y el de recepción), extrayendo a una hoja de cálculo los parámetros necesarios. Represéntelos gráficamente frente al tiempo y explique a qué se debe la existencia o no de regiones bien diferenciadas. Realice una segunda gráfica relativa a los tiempos entre llegadas y explique igualmente los resultados.
   2.3. (2 puntos) Mida a partir de los datos capturados a la salida del emulador, el caudal efectivo en bits/s que finalmente se obtiene tras pasar por el *token bucket*. Utilice para ello la estimación de ancho de banda medio aplicada en la Práctica 2. Compare el resultado obtenido con el previsto en el Apartado 1.
3. Utilice la Práctica 1 para estudiar la influencia del token-bucket sobre un flujo multimedia. Mediante dos VLC (emisor y receptor), envíe el vídeo proporcionado en dicha práctica a través del `emuladorTB.py`, para el escenario que coincide con su número de pareja. Recuerde que dicho vídeo es de tasa variable, con lo que será normal la presencia de picos en el mismo.
   3.1. (1 punto) Capture con Wireshark el tráfico y represente con la función IOgraph el caudal consumido por el vídeo a lo largo del tiempo, tanto a la entrada como a la salida del emulador (ambos flujos se pueden diferenciar fácilmente según su

puerto de destino). Explique la diferencia entre uno y otro flujo multimedia.

3.2. (1 punto) Cuantifique experimentalmente qué retardo se añade en el caso peor a los paquetes en la traza capturada. Wireshark posee una funcionalidad de análisis de flujos RTP que puede serle de utilidad en este caso.

3.3. (0,5 puntos) Estudie desde un punto de vista subjetivo si aprecia alguna merma de calidad percibida (MOS), y si es posible y de qué manera corregir este problema en el receptor.

4 Entrega

El trabajo realizado se entregará vía Moodle el día anterior al examen de la presente práctica. Puede consultar el calendario de prácticas en Moodle.

El archivo a enviar será un zip, que incluya la memoria de la práctica realizada en formato PDF, y los códigos implementados, incluyendo asimismo un archivo readme.txt que indique qué contiene cada archivo.

El zip entregado deberá tener el nombre con el formato:

RM-G-<Lunes/Martes/Jueves>-<número de grupo>-P4

Por tanto, si una pareja pertenece al grupo del Lunes y su pareja es la 03, el nombre del fichero de entrega de prácticas deberá ser:

RM-G-Lunes-03-P4.zip

5 Valoración de las prácticas

Para evaluar esta práctica se han definido los criterios que se presentan a continuación:

5.1 Entrega

La memoria debe contener toda la información necesaria para valorar el trabajo realizado en cada uno de los apartados propuestos en esta práctica. Igualmente, el código presentado debe contener la funcionalidad solicitada. La calificación será numérica de 0 a 10.

5.2 Evaluación de los conocimientos aprendidos

Los estudiantes serán evaluados tras la entrega respecto de los resultados obtenidos en la práctica, cubriendo dicha evaluación la totalidad de apartados de la misma.