REDES MULTIMEDIA **OpenCourseWare**

Práctica 4

Corrección de errores en la recepción de secuencias multimedia

1. Objetivos

El objetivo de esta práctica final es modificar un sencillo software para proteger una secuencia de paquetes a transmitir por un canal con pérdidas. El receptor conoce el esquema de protección y debe, en la medida de lo posible reconstruir la secuencia de paquetes enviada. En el caso de que las pérdidas superen las capacidades del sistema de protección, el receptor no podrá reconstruir correctamente la secuencia. El alumno, finalmente deberá dar una medida de la capacidad de protección de su sistema.

2. Introducción

Como ya se ha visto tanto en las clases de teoría como en las sesiones de prácticas, los flujos de datos multimedia se pueden transmitir usando el protocolo de transporte UDP. Este protocolo no dispone de mecanismos que aseguren la conexión, la protección frente a pérdidas de paquetes o la desorganización de la secuencia de los mismos en recepción.

El funcionamiento habitual de las aplicaciones reproductoras de voz que no utilizan ningún mecanismo de buffering o de corrección de errores, ya sean síncronas o asíncronas, es descartar los paquetes que lleguen tarde respecto a las marcas de tiempo incorporadas en el mismo o desordenados, elevando así la tasa de errores percibida por el receptor.

Existen diferentes mecanismos de protección frente a los efectos típicos que provoca la transmisión de los datos a través de una red de paquetes best-effort. En definitiva, lo que **nos va a preocupar en esta práctica es que se pierda la continuidad en la serie de paquetes que llegan al receptor**, ya sea por pérdidas o por desorden. De esta manera, podremos implementar nuestro algoritmo independientemente del sistema de protección o de almacenamiento de que disponga el reproductor del equipo receptor.

El código de la aplicación puede modificarse y compilarse de la forma que el alumno crea más conveniente en función de sus conocimientos previos, ya sea mediante un editor de texto y usando la herramienta de compilación javac de Linux, como con cualquier entorno de desarrollo que esté acostumbrado a utilizar.

Es recomendable familiarizarse con la ejecución remota de programas, tanto de visualización y monitorización de tráfico como de su propio software. Los comandos básicos pueden consultarse en Internet en una gran cantidad de enlaces, en concreto, para ejecutar el programa vlc en remoto y para automatizar las ejecuciones mediante scripts pueden ser útiles las siguientes referencias:

- Manual de VideoLan VLC: http://www.videolan.org/doc/streaming-howto/en/ch07.html

Así mismo, para todo el desarrollo de la práctica es recomendable consultar el API de la versión 1.8 de Java y del Java Media Framework (JMF), así como algunos libros útiles a la hora de programación de aplicaciones java en red.

En cuanto a los conocimientos teóricos, todos ellos forman parte de los temarios de asignaturas correspondientes al plan de estudios de la titulación, pudiendo consultar el alumno los apuntes de las mismas. Como referencia textual pueden usarse además de las referencias básicas de la asignatura, los siguientes textos básicos:

- *"Redes de computadoras"* Tanenbaum, Andrew S.
- *"Data and computer communications"* Stallings, William.
- *"TCP-IP illustrated"* Stevens, W. Richard.

3. Tareas Previas

Antes de comenzar la práctica es recomendable que el alumno:

- Refresque sus conocimientos de programación en java.
- Consulte el código facilitado antes de empezar a realizar la práctica.
- Consulte el funcionamiento de "synchronized" en java.
- Se asegure de entender correctamente el funcionamiento de los bloques emulador.

4. Descripción de un bloque emulador

El código fuente de un bloque de emulador puede descargarse de la plataforma OCW (emulador.zip). Para compilar el código fuente, simplemente necesitamos(desde el fichero emulador):

*javac *.java*

Para ejecutar esos ficheros, tenemos que cambiar al directorio padre de "emulador" y ejecutar el siguiente código:

java emulador.Emulador emulador/config.properties

Aunque lo vamos a ver más adelante, podemos probar a ejecutar el código de emulador como una caja negra que capturará tráfico UDP en un determinado puerto, lo procesará y lo enviará a un destinatario. La Figura 1 muestra el esquema de un montaje de red usando un bloque emulador.

El programa emulador, además de procesar el tráfico, nos informará de la cantidad de paquetes que recibe, que envía, el estado de las colas y de las pérdidas registradas. Internamente, cada bloque emulador está compuesto por una entidad receptora y por una emisora y caracterizado por un ancho de banda y un tamaño de cola. Todo ello ha de ser configurado mediante una serie de parámetros que podemos encontrar, y debemos configurar, en el archivo de configuración config.properties. Sólo una configuración correcta de dichos parámetros resultará en una comunicación satisfactoria. Los parámetros a configurar en el fichero son:

Figura 1. Esquema de envío a través de un emulador

- Puerto_in = puerto de escucha del bloque receptor del programa emulador.

- IP_destino = direccion IP del destino/receptor al que ha de ser enviado el tráfico.
- Puerto_out = puerto en destino al que se conecta el bloque emulador para enviar el tráfico.
- BW = Ancho de banda del sistema emulado.
- Queue_size = tamaño de la cola de espera.
- Time = intervalo de impresión de datos en pantalla

Los parámetros P_{in}, $IP_{Destino}$ y P_{out} aparecen en la figura en los bloques VLC porque están íntimamente relacionados con ellos. P_{in} es el puerto que configuramos en el VLC emisor como puerto de destino. $IP_{Destino}$ y P_{out} son la IP del VLC receptor y el puerto donde estará esperando la comunicación.

Los parámetros BW y Queue_Size permiten configurar la red que el bloque emulador está simulando y que se caracteriza por un tamaño de cola de espera y un ancho de banda. De esta forma, aunque estemos realizando nuestras pruebas en local en un mismo PC o entre dos PC's conectados por una LAN Ethernet, podremos emular el comportamiento de sistemas cuya red de acceso no tengan unas prestaciones tan buenas, por ejemplo, reduciendo el ancho de banda disponible y/o saturando la cola de entrada de los routers de acceso. El tamaño de la cola de espera será el número de paquetes máximo que aceptaría el buffer de entrada y que permanecen almacenados mientras se transmiten los que les preceden. En caso de saturación de este buffer, los paquetes que se reciben se descartan.

Para los objetivos de esta práctica no es necesario cambiar estos valores dinámicamente, los cambiaremos de manera manual eligiendo los valores adecuados conforme a los datos que vayamos a enviar.

El ancho de banda simulado se consigue mediante un rústico y sencillo cálculo del tiempo de transmisión necesario para cada paquete que se envía en una red con una velocidad de funcionamiento menor. Esto supone que en la realidad, nuestro programa funcionará como un sencillo conformador de tráfico. Los efectos de este conformado pueden observarse en wireshark.

5. Etapas de la práctica

5.1. Analizar el código fuente proporcionado

Análisis del código fuente de la aplicación para comprenderlo mejor. El alumno deberá analizar el código proporcionado (al menos las clases Emulador, Transmisor y Receptor) para entender el flujo de la aplicación. **Realizar un esquema del programa que facilite su comprensión por parte del alumno.**

5.2. Establecer un streaming de vídeo entre dos VLCs a través de un bloque emulador

Compilar el código e insertarlo entre el emisor y receptor VLC como si se tratase de un router en el camino. Se recomienda jugar con los parámetros de transmisión para conseguir una mejor comprensión del emulador. **Profundizar en los detalles del proceso de conexión.**

5.3. Implementación de un algoritmo de corrección de errores

El alumno tendrá que diseñar e implementar un algoritmo de protección frente a errores debidos a la pérdida de paquetes durante la transmisión en red. Esto es, implementar cada una de los bloques emulador necesarios y por último establecer un streaming de vídeo para comprobar su correcto funcionamiento.

El algoritmo de protección consistirá en añadir una redundancia de factor K, es decir, en repetir el paquete a transmitir K veces (por defecto elegiremos un factor K=4). El sistema que tendremos que configurar estará, por tanto, compuesto de un bloque transmisor y un bloque receptor. El bloque transmisor se encargará de crear la redundancia mientras que el receptor deberá eliminar las copias que no necesite de entre los paquetes que le llegan. Igualmente, es importante que el alumno medite si dicha adición de redundancia, o eliminación de la misma, se va a hacer en la parte receptora o emisora del bloque emulador. El sistema completo quedará como se muestra en la Figura 2. **OPCIONAL:** *Ajustar el código para que K sea configurable desde el fichero de configuración pasado como parámetro*

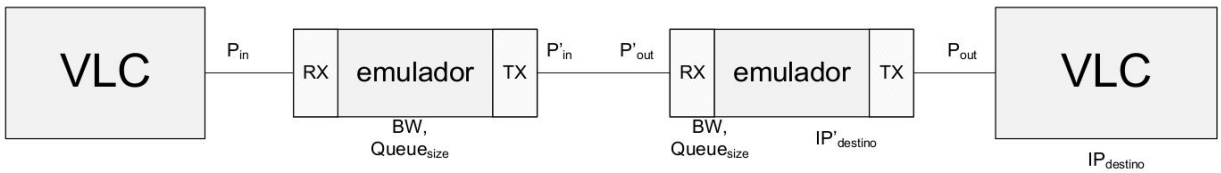

Figura 2. Esquema completo del sistema de corrección de errores

Tenga en cuenta que es imprescindible una correcta configuración de la cadena de IP/puertos de entrada y salida de cada uno de las cuatro entidades involucradas. Igualmente, piense que no es necesario modificar el contenido de los paquetes de un stream para que estos sean tratados correctamente en el receptor, sino simplemente leer la información de las cabeceras ya que en los paquetes replicados el número de secuencia interno será el mismo.

El código disponible para el alumno se corresponde a uno de los bloques emulador, por tanto, para la realización de este apartado será necesario configurar dos bloques distintos e independientes. Cada uno de ellos tendrá que ser configurado de manera distinta atendiendo a su función. Se recomienda tratar cada uno de estos bloques como un proyecto distinto, teniéndolos en distintos árboles de carpetas y encapsulándolos en distintos paquetes (línea package al principio de cada uno de los archivos de código).

Para comprobar el funcionamiento de su sistema de corrección de errores, es necesario que los paquetes de video se manden desde el VLC mediante RTP. De esta forma, los paquetes tendrán su correspondiente cabecera, en la que, como se puede ver en la figura, el número de secuencia del paquete, que es la información relevante para este objetivo, se encuentra en las posiciones 2 y 3 del array de bytes correspondiente a los datos del paquete. La información correspondiente a este protocolo está disponible en la RFC 3550 del IETF[1]. La Figura 3 muestra el contenido habitual de un paquete RTP.

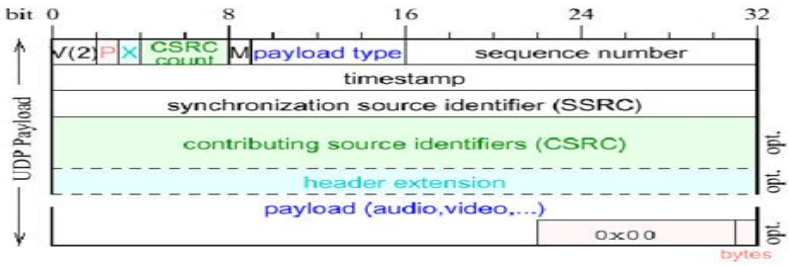

Figura 3. Cabecera de un paquete RTP

5.4. Comprobación de las prestaciones del algoritmo

En este apartado comprobaremos la capacidad del software implementado por el alumno para lidiar con errores en la red. Por ello introduciremos un nuevo bloque intermedio en el sistema previo que será el encargado de introducir pérdidas en el sistema. Esto lo podemos conseguir, principalmente, de dos maneras, bien (1) modificando los parámetros de ancho de banda y de tamaño de las colas del bloque añadido, o bien (2) forzando pérdidas con una determinada distribución de errores en función de la cual los paquetes se reenviarán por dicho bloque o serán descartados. El esquema de bloques para este sistema se muestra en la Figura 4.

[1] Disponible en: http://www.ietf.org/rfc/rfc3550.txt

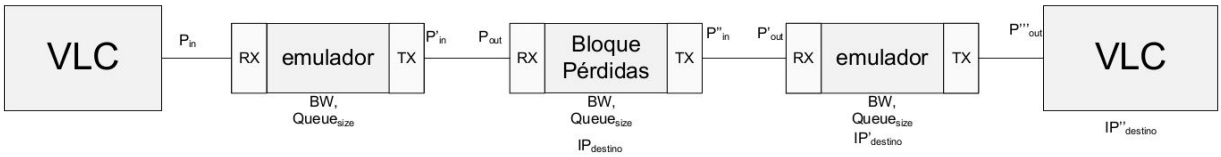

Figura 4. Sistema con el emulador generador de pérdidas

En este apartado, el alumno deberá implementar dos distribuciones de error de manera obligatoria:

- Una distribución uniforme
- Una distribución exponencial

Para configurar estas distribuciones habrá que añadir uno (o dos, en función de la distribución) parámetros al archivo de parámetros, la media y la desviación. Además tendrá que utilizar otro parámetro indicando cual es la distribución que se ha de usar. Dichos parámetros deberán ser interpretados por el bloque emulador. **OPCIONAL:** *Añadir alguna otra distribución de pérdidas, como por ejemplo, la gaussiana*

Una vez implementadas estas distribuciones, jugaremos con la probabilidad de pérdidas. Comenzando por 0%, 5%, 10% e incrementando con saltos de un 10% hasta un 80%. Al aumentar el porcentaje de pérdidas llegará un momento en que nuestro algoritmo no sea capaz de evitar las pérdidas. Llegado a este punto, anote el porcentaje de pérdidas registrado para la media de errores configurada en su distribución. **OPCIONAL.** *Estudie el desempeño del algoritmo para distintos valores de K.*

6. Aspectos importantes de cara a la realización de la práctica

A parte de la implementación del sistema de redundancia, hay que implementar las funciones que nos permitan introducir pérdidas en el sistema. Pese a que para la distribución uniforme no hace falta ningún tipo de función especial, no ocurre lo mismo para la implementación de la distribución normal o de la exponencial. Para ello se incluye un paquete JAR con las librerías "Common Math" de apache que se recomienda al alumno usar, ya que contienen funciones que simplificarán mucho la implementación de estas funciones. Obviamente, para usar este paquete habrá que saberlo llamar desde el código mediante la instrucción import oportuna.

Otra parte importante de la práctica es el ajuste de los parámetros de cada uno de los bloques emulador para poder transmitir y recibir correctamente las secuencias de vídeo. Parámetros importantes para esto son, a parte de los ya mencionados ancho de banda y tamaño de las colas, los tamaños de los buffers de los sockets de la parte de recepción y emisión de los bloques emulador. Igualmente, se recomienda al alumno fijarse en las estadísticas mostradas por estos bloques en cuanto a paquetes emitidos y recibidos, ya que esto le ayudará a saber cuándo ha de aumentar el caudal de alguno de los bloques porque éste esté emitiendo a una tasa muy baja o porque esté habiendo algún tipo de problema en él. Si el tamaño de los buffers es muy pequeño estaremos limitando en origen o destino nuestra velocidad de emisión/recepción y no podremos establecer una comunicación satisfactoria.

Finalmente, será también importante tener en cuenta como se manejan los bloques synchronized que tenemos en los bloques trasmisor y emisor ya que manejarlos de una manera incorrecta puede resultar en retrasos a la hora de hacer una u otra operación.