Redes Multimedia

Práctica 1: Introducción a redes multimedia

Turno y pareja: 2461-06

Integrantes:

Pablo Diez del Pozo

Alejandro Alcalá Álvarez

Fecha de entrega: 24 de Febrero de 2020

Contenido

[1Introducción 2](#__RefHeading___Toc677_1860594804)

[2Realización de la práctica 2](#__RefHeading___Toc679_1860594804)

[3Conclusiones 5](#__RefHeading___Toc683_1860594804)

# Introducción

Hemos realizado un emulador que reciba los paquetes multimedia por un puerto y los retransmite por otro puerto a un reproductor multimedia, en nuestro caso es VLC. El emulador que vamos a implementar va a tener como la función que tienen la red entre un servidor y un cliente, el cual quiere obtener recursos multimedia del servidor. El emulador añadirá un retraso a los paquetes llamado *jitter* y también tendrá una probabilidad de perdida del paquete.

# Realización de la práctica

## 

1. Se realizará un módulo emulador que reciba paquetes multimedia por un puerto y los retransmita a un destino en otro puerto. Dicho módulo simulará comportarse como la red que hay habitualmente entre un servidor y un cliente. Para implementarlo, puede aprovechar el código de la práctica anterior modificándolo adecuadamente.

Para realizar esta simulación, el emulador añadirá un retardo variable (jitter) y una probabilidad de pérdida a cada paquete recibido a la hora de retransmitirlo al destino final. Estos valores deberán ser parametrizables como argumentos de entrada al programa:

* 1. Direcciones y puertos donde escucha el emulador, y a dónde se enviarán finalmente los paquetes.
  2. Probabilidad de pérdida de paquete (valores entre 0 y 1).
  3. Rango de variación del retardo, suponiendo una distribución uniforme.

Por ejemplo, para ejecutar el programa, deberá escribir:

python emulador.py *<dirección\_escucha> <puerto\_escucha> <dirección\_destino> <puerto\_destino> <pérdida> <retardo\_mínimo\_en\_ms> <retardo\_máximo\_en\_ms>*

Y un ejemplo de valores sería:

python emulador.py 127.0.0.1 5002 127.0.0.1 5004 0.01 0 100

|  |
| --- |
| Hemos tenido que importar las librerias sys, socket, random y time. La librera sys la hemos importado para obtener los argumentos de entrada que tiene nuestro emualdor. La librería socket la hemos importado debido a que es el núcleo de la funcionalidad de nuestro porgrama, debido a que nuestra comunicación entre nuestro servidor y cliente va ser por sockets. La librería random se importa debido a que tenemos que generar números aleatorios para ver si hay perdida de paquetes y para ver el retraso que le añadimos al paquete. La librería time se añade para que el programa espere un tiempo determinado antes de enviar el paquete al cliente. Abrimos por lo tanto dos socket, donde uno de ellos es para escuchar los paquetes que le llegan del servidor multimedia y el otro socket es para enviar los paquetes al cliente multimedia. El socket con el que escuchamos al servidor multimedia hacemos un bind para enlazarlo a la IP y al puerto que nos han pasado por parámetro del programa. Después de enlazar el socket de escucha, aumentamos el tamaño del buffer para que reciba todos los paquetes completos y no tengamos perdidas de información en los paquetes. Creamos un bucle while donde vamos a obtener todos los paquetes que quiere enviar el servidor multimedia al cliente y cuando obtenemos esos paquetes tenemos que añadirle jitter a ese paquete o podemos desechar ese paquete dependiendo de los argumentos que le hayas introducido en el programa. Este emulador se pone entre medias de dos reproductores multimedia llamado VLC, el cual tiene una función para poder emitir y recibir contenido multimedia. |

1. Utilice el programa VLC para enviar un vídeo al puerto en que escucha el emulador, y otro VLC para recibir el vídeo desde el puerto de destino al que envía el emulador. Indique los parámetros que utiliza para emitir el vídeo y para recibirlo, explicando qué posibilidades existen en cada caso, y el motivo de las escogidas.

|  |
| --- |
| Los parámetros que se utilizan para emitir y recibir son los cuatro primero. Es decir, la dirección IP fuente y puerto fuente que están asociados al primer VLC debido a que es el que utilizamos para enviar el contenido multimedia y en e otro VLC tenemos que configurarlos con la dirección IP destino y el puerto destino. A continuación mostraremos un ejemplo de como configurar los VLC para poder utilizar el programa.  1. Abrimos el VLC y damos a la pestaña de medio y seleccionamos la opción de Emitir y nos saldrá esta ventana:    2. Le damos al botón de añadir y nos saldrá una ventana para buscar el archivo que queremos emitir, cuando lo tengamos seleccionamos damos al botón de Emitir y nos llevará a la siguiente ventana:    3. A continuación, le daremos a Siguiente y nos saldrá la siguiente ventana, que tendremos que pinchar en RTP/MPEG transport Media y darle a añadir. Después de añadir debemos introducir una dirección y un puerto, esto son los parámetros que le tenemos que meter por entrada a nuestro programa y le daremos a siguiente después de introducir estos parámetros:    4. Después, nos saltará otra pantalla donde tenemos que poner como queremos que vaya codificado el video por la red y le daremos a siguiente para saltar a la siguiente ventana:    5. Después de confirmar como queremos codificar el video, saltaremos a la siguiente pestaña y le daremos a emitir, debido a que ya hemos configurado el primer VLC para emitir contenido multimedia:    A continuación, explicaremos los pasos que tenemos que hacer para la configuración del segundo VLC.  1. Abrimos el VLC, pinchamos en la pestaña de “Medio” y seleccionamos la opción de “Abrir ubicación de red”, donde nos saldrá esta ventana:    2. En la ventana anterior tenemos que introducir la siguiente cadena en la URL: “rpt://dirección\_origen:puerto\_origen”. Por último, pinchamos en “Reproducir” y desde ese momento el VLC se quedará esperando a recibir paquetes del proxy. |
| Existe distintas posibilidades de envio del video entre los dos VLC que son las siguientes posibilidades: |
| La escogida por nosotros es RTP/MPEG Transport Stream debido a que es el más adecuado para la realización de las pruebas que tenemos que hacer con los distintos emuladores que tenemos que realizar. Otro motivo por a escogida de este tipo es que las demás opciones no son recomendadas para en envio de videos por streaming. |

1. Desarrolle una segunda versión del módulo (emulador2.py) para que el retardo introducido a un paquete no dependa del paquete anterior, utilizando para ello *threads*. Explique qué resuelve con este cambio.

|  |
| --- |
| Para esta nueva versión del emulador, hemos creado una función a parte que calcule los retrasos y envie el paquete al segundo VLC. Los detalles de la función que hemos implementado es la siguiente:    Donde le introducimos los siguientes parámetros:  - El socket creado para el cliente para enviarle los paquetes del primer VLC.  - Los datos que tenemos que enviar al cliente.  - El mínimo de retraso que puede haber en el envio de un paquete.  - El máximo de retraso que puede haber en el envio de un paquete.  - La IP del cliente al que queremos enviar el paquete.  - El puerto del cliente al que queremos enviar el paquete.  En nuestra función principal hemos declarado un hilo que va ejecutar la función anterior y le pasamos por argumento todos los valores que nos han introducido por linea de comando o que nos ha enviado el primer VLC. Con esta función creamos un hilo por cada paquete que recibamos desde el primer VLC. Después de decidir si desechamos el paquete o no, iniciamos la ejecución del hilo con el método “start” de la clase Thread. |

1. Estudie la diferente degradación que se produce con la introducción de jitter y de pérdidas en la transmisión del contenido multimedia, valorándola de acuerdo a la escala MOS (*Mean Opinion Score*, Puntuación de Opinión Promedio):

|  |  |  |
| --- | --- | --- |
| **MOS** | **Calidad** | **Degradación** |
| **5** | Excelente | Imperceptible |
| **4** | Buena | Perceptible pero no molesta |
| **3** | Justa | Ligeramente molesta |
| **2** | Pobre | Molesta |
| **1** | Mala | Muy molesta |

Utilice para ello los siguientes valores de pérdida de paquetes y de *jitter*:

* Porcentaje de pérdida de paquetes: {0;0,2; 0,5; 0,9; 2; 5; 9}%,
* Retardo mínimo y máximo: {(0, 0); (0, 250); (0, 500); (0,750); (0, 1000)} (ms)

En caso de que no observe variaciones sustanciales con algunos de los valores anteriores, indique para qué valores ha observado variaciones en el MOS.

Rellene la tabla siguiente con los resultados para el integrante 1 de la pareja.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Retardo/Pérdidas | 0 | 0,2% | 0,5% | 0,9% | 2% | 5% | 9% |
| (0, 0)ms | 5 | 4 | 4 | 3 | 1 | 1 | 1 |
| (0, 250) ms | 5 | 3 | 2 | 2 | 1 | 1 | 1 |
| (0, 500) ms | 5 | 3 | 3 | 2 | 1 | 1 | 1 |
| (0, 750) ms | 5 | 3 | 2 | 1 | 1 | 1 | 1 |
| (0, 1000) ms | 5 | 3 | 2 | 1 | 1 | 1 | 1 |

Rellene la tabla siguiente con los resultados para el integrante 2 de la pareja.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Retardo/Pérdidas | 0 | 0,2% | 0,5% | 0,9% | 2% | 5% | 9% |
| (0, 0)ms | 5 | 5 | 4 | 3 | 2 | 1 | 1 |
| (0, 250) ms | 5 | 5 | 3 | 3 | 1 | 1 | 1 |
| (0, 500) ms | 5 | 5 | 4 | 3 | 2 | 1 | 1 |
| (0, 750) ms | 5 | 4 | 3 | 2 | 1 | 1 | 1 |
| (0, 1000) ms | 5 | 4 | 3 | 3 | 1 | 1 | 1 |

Rellene la tabla siguiente con los resultados promediados entre ambos integrantes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Retardo/Pérdidas | 0 | 0,2% | 0,5% | 0,9% | 2% | 5% | 9% |
| (0, 0)ms | 5 | 4,5 | 4 | 3 | 1,5 | 1 | 1 |
| (0, 250) ms | 5 | 4 | 2,5 | 2,5 | 1 | 1 | 1 |
| (0, 500) ms | 5 | 4 | 3,5 | 2,5 | 1,5 | 1 | 1 |
| (0, 750) ms | 5 | 3,5 | 2,5 | 1,5 | 1 | 1 | 1 |
| (0, 1000) ms | 5 | 3,5 | 2,5 | 2 | 1 | 1 | 1 |

1. Tras el estudio anterior, valore qué problema considera más molesto desde el punto de vista del usuario, y cuál cree que puede darse más frecuentemente en la red. Razone hasta qué punto las pérdidas y retardos propuestos son habituales o no en la red.

|  |
| --- |
| Después de todas las pruebas realizadas, nosotros consideramos que nos resulta más molesto la combinación del jitter con la perdida de paquetes. Debido a que una combinación perdemos el hilo del video y del audio, por lo tanto, no tendriamos una visión correcta del video y podemos perdernos cosas importantes del video y perdemos también calidad en la imagen del video. Nosotros consideramos que en la realidad hay mucho más jitter que perdida de paquetes por la red. |
| Para los retrasos depende mucho de la saturación de la red en ese momento, pero un jitter de un segundo nos parece excesivo que se de en la realidad.  Para la perdida de paquetes también depende mucho de la saturación de la red y el reproductor, el cual vayas a reproducir el streaming, debido a que VLC tiene un buffer de un 1 segundo, es decir, si algún paquete que llegue al buffer y este lleno este se descarta. Por lo tanto, el rango de tener un 10% de perdidas me parece muchismo y es muy raro que se produzca en la realidad. |

1. Indique de manera aproximada cuánto tiempo tarda inicialmente el VLC en comenzar la reproducción del vídeo, e indique a qué se debe.

|  |
| --- |
| Nuestro segundo VLC tarda un tiempo aproximado de 3-4 segundos en iniciar la retrasmisión del video, debido a que el VLC tiene que conseguir almacenar los paquetes suficientes en su buffer para iniciar la transmisión. |

1. Varíe los parámetros de que dispone VLC para compensar el *jitter*. Indique qué valores modifica y qué impacto tienen en la calidad percibida. Explique si considera adecuados estos valores para una comunicación interactiva en la que dos usuarios quieran mantener una videoconferencia.

|  |
| --- |
| El principal parámetro que tiene el VLC es el buffer de recepción, este buffer nos permite especificar el retraso que pueden tener los paquetes desde el primer paquete hasta que se inicie la reproducción del video. Por lo tanto, cuando comienza el video a retransmitirse en el segundo VLC, el buffer de recepción de ese VLC ha obtenido los suficientes paquetes para que no se note el jitter en la reproducción del video multimedia. |
| No es del todo adecuado, debido a que queremos tener una conversación a tiempo real y si utilizamos el buffer siempre vamos a ir con retraso a lo que diga la otra persona con la que estamos hablando. El buffer de recepción no sería la solución más adecuada para tener una conferencia fluida entre dos personas. |

1. Capture el tráfico y analice cómo varían los tiempos entre llegadas en el emisor y el receptor para el caso de *jitter* máximo. Obtenga un histograma para ambos casos y evalúe la influencia del *jitter*.

|  |
| --- |
|  |
|  |
| **¿Qué diferencia observa entre ambos histogramas? ¿Qué influencia tiene el jitter?**  Podemos observar que en el emisor todos los paquetes están en el rango de tiempo de 0 y 0,0125 y en el segundo histograma la mayoría de los paquetes esta en el rango entre 0 y 0,025. Tiene una gran influencia debido a que tenemos grandes retrasos entre lass llegadas de los distintos paquetes que enviamos del primer VLC al segundo VLC. |

1. Genere una tercera versión del módulo (emulador3.py) para que la variación del retardo siga una distribución gaussiana con un valor medio y una cierta varianza. Tenga en cuenta que el valor del retardo como mínimo debe ser 0. Los parámetros de entrada de este tercer módulo deben ser iguales a las dos anteriores.

|  |
| --- |
| Para obtener los parámetros para ejecutar el método gauss de random, hemos utilizado la regla de las tres sigma. Para encontrar la media, hemos hecho la siguiente formula:  Media = (max\_delay – min\_delay) / 2  Y para calcular la varianza de los retrasos, hemos utilizado la siguiente formula:  sigma = (media – min\_delay) / 3  Por lo tanto, ahora al hilo le tenemos que pasar dos argumentos más que son la media y sigma que hemos calculado en programa principal. Tambien le pasamos el max\_delay y el min\_delay debido a que el método puede haber una probabilidad de que caigan valores fuera de ese intervalo, si en el caso que cayera fuera de ese intervalo volver a calcular el número hasta que caiga dentro del intervalo. |
| **¿Qué valores de media y desviación ha utilizado a partir del retardo mínimo y máximo introducidos por línea de comando?**  Hemos utilizado el máximo jitter que podemos tener en la realidad, es decir, de valor mínimo hemos introducido 0 y de valor máximo hemos introducido el valor 1000 ms (1 s). Los valores de la media ha sido de 500 ms (0,5 s) y el valor de la desviación que hemos tenido ha sido de 166,67 ms (aproximadamente de 0,167s).  A continuación, os vamos a mostrar los diferencias entre la recepción de paquetes en el emisor y del receptor del emulador 2 y del emulador 3: |
| **¿Qué diferencias aprecia con respecto al retardo que sigue una distribución uniforme en el experimento del MOS para jitter máximo? ¿A qué se debe?**  Tenemos mejores resultados en el experimento del MOS, debido a que los retrasos que hay entre paquetes no es tan variado y siempre coge un valor muy proximos a la media del intervalor cogido y la uniforme tiene la misma probabilidad de coger valores pequeños de jitter como valores muy proximos al jitter máximo. |

# Conclusiones

Las conclusiones que hemos obtenido en esta práctica, es ver lo difícil que es la transmisión de archivos multimedias y los distintos problemas que puede tener el envío de estos archivos por la red. También hemos visto que podemos configurar el reproductor multimedia que utilicemos para poder mitigar el jitter y para que nosotros lo podamos ver en mejor condiciones y poder disfrutar de una calidad del video buena. También hemos visto que la realización de un proxy es mejor con hilos debido a que el envío de los paquetes no están asociados a la lógica de la aplicación y pueda calcularlos retrasos y hacer la espera paralelamente a la ejecución del porgrama principal.