Redes Multimedia – Prácticas 2019

Práctica 2: Medidas de rendimiento

Turno y pareja: 4312\_02

Integrantes:

Javier Martín González

Jorge Gutiérrez Díaz

Fecha de entrega: 18-03-2019

Contenido

[1Introducción 2](#__RefHeading___Toc285_732697610)

[2Realización de la práctica 2](#__RefHeading___Toc287_732697610)

[3Conclusiones 5](#__RefHeading___Toc289_732697610)

# Introducción

Escriba aquí una introducción al trabajo realizado en la práctica.

# Realización de la práctica

## 

1. Se entrega un programa que envía trenes de paquetes en los que se puede configurar la longitud del tren y la longitud del campo de datos de los paquetes por encima del nivel de aplicación. Los paquetes poseen un campo de número de secuencia y otro de marca de tiempos. Para ello se utiliza la cabecera típica de RTP, según se proponía en la práctica 0. Los parámetros se entrarán por línea de comandos siguiendo la sintaxis:

*clienteTren.py ip\_destino puerto\_destino longitud\_tren longitud\_datos*

Estudie en detalle el funcionamiento del programa, pues deberá modificarlo en un apartado posterior. ¿Qué valor de marca de tiempo se envía? ¿Qué relación tiene con el tiempo *epoch* (segundos desde 1970)?

|  |
| --- |
| En primer lugar se realiza una comprobación de la cantidad de parámetros introducidos. En el caso de que se haya introducido un número incorrecto de parámetros, el programa devuelve un mensaje mostrando como iniciar el cliente de forma correcta.  A continuación, el programa recoge los parámetros y comprueba que el tamaño de datos sea lo suficientemente grande como para que pueda ser enviado y lo suficientemente pequeño como para que no se produzca fragmentación.  Por último, añadimos a la cabecera RTP de cada paquete un número de secuencia que empieza desde 0 y aumenta con cada paquete, un identificador común para cada paquete que sea la longitud del tren a enviar, y el tiempo en que se envió cada paquete (segundos y centésimas de segundo) dentro del timestamp. |
| Dentro de la cabecera RTP, el valor del timestamp se puede modificar tanto en segundos como en centésimas de segundos truncados a 32 bits. En nuestro caso, obtenemos el tiempo del envio del paquete en segundos y lo multiplicamos por la constante DECENASMICROSECS para pasarlo a decenas de microsegundos. Por último, el resultado lo pasamos a través de una máscara para truncarlo a 32 bits. |
| El tiempo *epoch* es el tiempo transcurrido desde el 1 de enero de 1970 y que dura hasta nuestros días. El tráfico en la red utiliza este formato de tiempo para reflejar las distintas marcas de tiempo, por lo tanto, utilizamos este formato para imitar el tráfico real de la red. La función de time.time() devuelve el tiempo en segundos desde el tiempo *epoch* como un numero de punto flotante. |

1. Se debe completar un programa que recibe los trenes de paquetes, de forma que mida y visualice por pantalla anchos de banda (instantáneos, máximo, medio y mínimo), retardos en un sentido (instantáneos, máximo, medio y mínimo), variación del retardo, y pérdida de paquetes (%).Tenga en cuenta que los paquetes que se envíen a la red contendrán igualmente las cabeceras de las capas inferiores (RTP, UDP, IP, Ethernet) según se mostraba en la Error: no se encontró el origen de la referencia, por lo que dicha longitud también debe ser tenida en cuenta a la hora de medir el ancho de banda. En el caso de utilizar la interfaz local no habrá cabecera Ethernet. El servidor deberá ejecutarse según la siguiente sintaxis:

*servidorTren.py ip\_escucha puerto\_escucha*

|  |
| --- |
| Se han creado tres listas para guardar el ancho de banda, el retardo de los paquetes y el tiempo de llegada de estos. Como las mediciones pueden realizarse en localhost o entre dos equipos, se ha utilizado un *if* que sume al tamaño de cabeceras totales la cabecera de *ethernet* en el caso que se haga las mediciones entre dos equipos.  A la llegada de un paquete, se guarda el retardo y el tiempo de llegada del paquete en sus respectivas listas. A partir del segundo paquete recibido, se calcula el ancho de banda instantáneo sumando el tamaño de datos del paquete con el tamaño total de las cabeceras y se multiplica por 8 para pasarlo a bits. El resultado se divide entre la diferencia del tiempo de llegada del último paquete con el anterior, y el ancho de banda obtenido se guarda en la lista de anchos de bandas. El ancho de banda que se muestra por pantalla es el calculado pero dividido entre 1000000 para pasarlo a Mb/s.  Para el ancho de banda y retardo medio, máximo y mínimo se ha aplicado las funciones *statistics,mean, max* y *min* de listas para obtener los resultados respectivamente. Además, se ha calculado el factor de dispersión en el servidor como el tiempo transcurrido entre el último bit del primer paquete y el último bit del último paquete del tren.  Por último, se calcula el porcentaje de paquetes perdidos dividiendo la diferencia de la longitud del tren menos la cantidad de paquetes recibidos, entre la longitud del tren. Todo se ha multiplicado por 100 para obtener el porcentaje. El *jiiter* se obtiene con la varianza de la lista de retardos mediante la función de *statistics.variance.* |

1. Pruebe ambos programas en la interfaz local y entre dos equipos conectados en la red de área local. Realice varias medidas variando la longitud del tren y la longitud de los paquetes. Responda a las siguientes preguntas. Utilice el Wireshark en el receptor para ver los paquetes del tren y contrastar las respuestas.

|  |
| --- |
| Se utilizó el comando *ifconfig* en el equipo donde se ejecuta el servidor para obtener la *ip* de este*.* Se ejecuta el cliente desde un ordenador diferente y envia los paquetes a la ip donde se ejecuta el servidor. Wireshark corre en el mismo equipo que el cliente y filtramos los paquetes RTP con *ip* destino al equipo del servidor. |

* 1. ¿Qué valores ha empleado para contabilizar las distintas cabeceras?

|  |
| --- |
| IP = 20, ETH = 26, UDP = 8, RTP = 12 |

* 1. ¿Cuál es la longitud mínima de una de las tramas que se envían? ¿Por qué?

|  |
| --- |
| 6 bytes. El valor mínimo de datos que puede enviar *Ethernet* es de 46 bytes, y la suma de todas las cabeceras que se envían es de 40 bytes. Si se resta el valor mínimo de la cabecera Ethernet con la suma de las cabeceras, se obtiene que el valor mínimo de las tramas debe ser de 6 bytes. |

* 1. ¿Cuál es la longitud máxima de datos que tiene sentido utilizar? ¿Por qué?

|  |
| --- |
| Si se aplica la misma técnica pero con el cambio de que el valor máximo de la cabecera Ethernet es de 1500 bytes, se obtiene que el valor máximo de las tramas es de 1460 bytes. |

* 1. ¿Con qué longitudes de tren y datos se consiguen mejores resultados? ¿Por qué?

|  |
| --- |
| Se obtienen mejores resultados con longitudes de trenes grandes y tamaños de paquetes grandes porque así evitamos que el tráfico interferente de la red interfieran en los resultados. En nuestro caso, con trenes de 1000 paquetes y de 1000 bytes de datos se obtenían buenos resultados. |

1. Para poder medir adecuadamente retardos y *jitter* es necesario que el cliente envíe a una tasa inferior a la de la red, de forma que se elimine el efecto del cuello de botella sobre el tren. Modifique el programa proporcionado en el apartado para que permita configurar también la tasa de envío de los paquetes. Su sintaxis será:

*clienteTren2.py ip\_destino puerto\_destino longitud\_tren longitud\_datos [tasa\_binaria]*

Si no se indica la tasa binaria, se transmitirá a la tasa máxima posible, lo que permite hacer una estimación del ancho de banda, que se puede utilizar posteriormente para hacer adecuadamente las medidas de retardo y *jitter*.

|  |
| --- |
| Explique aquí el método empleado para limitar la tasa de transferencia del tren de paquetes. |

1. Realice medidas con clienteTren2.py y servidorTren.py, utilizando el emulador compilado que se puede descargar de Moodle. El emulador es un programa que simula retardos variables, pérdidas (similar al realizado en la práctica 1) pero también anchos de banda. Dicho ejecutable tiene la siguiente sintaxis:

*emulador ip\_escucha puerto\_escucha ip\_destino puerto\_destino DNI*

donde ip\_esucha y puerto\_escucha son la dirección IP y puerto donde escucha el emulador, ip\_destino y puerto\_destino son la dirección IP y puerto a donde el emulador reenvía lo que recibe, y DNI es un número de DNI (sin letra). El programa, siempre en base al número del DNI que se proporcione, impone una combinación de ancho de banda, retardo, variación del retardo y porcentaje de pérdida de paquetes único.

Deduzca a partir de las medidas realizadas qué valores de ancho de banda, retardo, variación del retardo y pérdidas se están aplicando en el emulador para el DNI de ambos miembros de la pareja. **Tenga en cuenta que para medir los retardos de forma correcta es necesario limitar la tasa de transferencia a una inferior o igual a la medida.**

|  |
| --- |
| Explique cómo ha realizado las medidas en cada caso. |
| DNI 1: número de DNI.  Ancho de banda estimado.  Retardo estimado.  Desviación estándar del retardo.  Porcentaje de pérdidas. |
| DNI 2: número de DNI.  Ancho de banda estimado.  Retardo estimado.  Desviación estándar del retardo.  Porcentaje de pérdidas. |

1. Capture el tráfico de las medidas realizadas con el emulador y analice con Wireshark el tráfico recibido y a partir de los tiempos de llegada, marcas de tiempo y longitudes, compare estos datos con los resultados obtenidos con su programa.

|  |
| --- |
| Explique cómo ha realizado las medidas en cada caso. |
| DNI 1: número de DNI.  Ancho de banda estimado.  Retardo estimado.  Desviación estándar del retardo.  Porcentaje de pérdidas. |
| DNI 2: número de DNI.  Ancho de banda estimado.  Retardo estimado.  Desviación estándar del retardo.  Porcentaje de pérdidas. |

1. Se desea establecer un servicio de VoIP sobre una red cuyos parámetros de calidad son los del emulador. Explique razonadamente qué códec y tiempos de paquetización deberá utilizar en ambos casos para adaptarse de la mejor manera posible al canal, y cuantas llamadas simultáneas se podrían soportar en ese caso (se supone una red full-duplex). Para valorar dicho códec y tiempo de paquetización puede utilizar [esta página](http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml). Igualmente, indique el tamaño del buffer a configurar en el receptor de la llamada para amortiguar el efecto del *jitter*.

|  |
| --- |
| Explique razonadamente el códec y tiempos de paquetización a utilizar para el canal del DNI1. |
| Indique cuántas llamadas se podrían soportar en este caso. |
| Indique razonadamente el tamaño del buffer a configurar en el receptor. |
| Explique razonadamente el códec y tiempos de paquetización a utilizar para el canal del DNI2. |
| Indique cuántas llamadas se podrían soportar en este caso. |
| Indique razonadamente el tamaño del buffer a configurar en el receptor. |

1. Simule el servicio planteado en el apartado 7 con las herramientas codificadas previamente y evalúe si el resultado responde a la predicción realizada. Indique razonadamente los parámetros utilizados para generar el tren de paquetes.

|  |
| --- |
| Indique razonadamente los parámetros utilizados para generar el tren de paquetes que simula el tráfico de VoIP calculado en el apartado anterior. |
| Evalúe con las medidas en el servidorTren si el resultado es |
|  |

# Conclusiones

Escriba aquí las conclusiones que ha extraído de la realización de la práctica.