Redes Multimedia
Práctica 2: Medidas de rendimiento

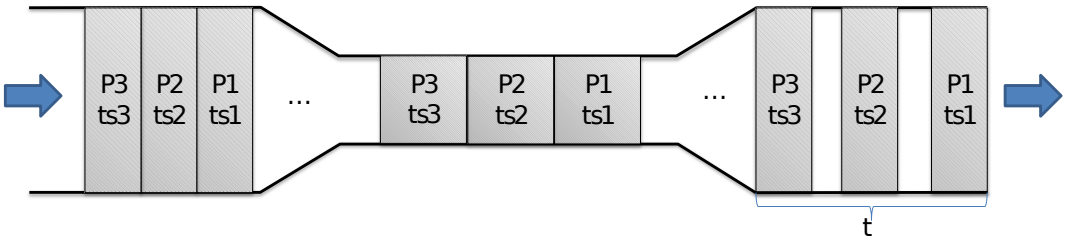
Contenido

1 Introducción

A la hora de desplegar un servicio multimedia sobre una red IP best-effort es importante conocer a priori la calidad que dicha red proporcionará, para configurar el servicio teniendo en cuenta el ancho de banda, las pérdidas o el retardo en un sentido extremo a extremo (entre el servidor y el cliente final). Por ejemplo, si el ancho de banda es bajo, será necesario seleccionar un códec de baja tasa; si el retardo en un sentido es alto, habrá que minimizar el tiempo de buffer, o reducir el número de muestras por paquete; si hay un número de pérdidas apreciable, habrá que establecer mecanismos de redundancia que permitan recuperar el contenido de paquetes perdidos.

Una técnica para realizar estas medidas se basa en el envío de trenes de paquetes desde un extremo al otro, pudiendo así el otro extremo medir los parámetros de calidad (ancho de banda, retardo en un sentido, pérdidas).

En lo que se refiere a medir el ancho de banda disponible, la técnica se basa en la siguiente idea: se envían varios paquetes seguidos a la tasa máxima posible (sin espacio entre ellos). Si en algún momento dichos paquetes pasan por un enlace que actúe como cuello de botella (un enlace de menor capacidad), los paquetes se recibirán con un espacio entre ellos que vendrá determinado según la capacidad de dicho cuello de botella.



Así, el ancho de banda (**R**) vendrá dado por el tamaño en bits (**L**) del tren de paquetes entre el tiempo que ha transcurrido entre la recepción del primer y último paquete del tren (**t**), según la fórmula **R=L/t** . Es posible igualmente calcular anchos de banda instantáneos entre pares de paquetes consecutivos siguiendo este mismo algoritmo.

Igualmente, a dichos paquetes se les puede añadir información, tal como un número de secuencia y una marca de tiempo, lo que permite igualmente medir pérdidas y retardo en un sentido (esto último siempre que los relojes del emisor y el receptor estén sincronizados, por ejemplo, utilizando NTP), así como la variación del retardo.

2 Objetivo de la práctica

El objetivo fundamental de esta práctica es introducir al alumno en las medidas de calidad en redes multimedia. Para ello, se pretende alcanzar los siguientes subobjetivos:

1. Conocer la técnica de medida basada en trenes de paquetes.
2. Valorar los parámetros de calidad de una red.
3. Aplicar las medidas realizadas para el despliegue de un servicio concreto.
4. Aplicar los conocimientos aprendidos en prácticas y asignaturas anteriores.

3 Realización de la práctica

1. (1 punto) Se realizará un programa que envíe trenes de paquetes en los que se pueda configurar la longitud del tren y la longitud del campo de datos de los paquetes por encima del nivel de aplicación. Los paquetes deben poseer un campo de número de secuencia y otro de marca de tiempos. Para ello se deberá utilizar 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
2. (1 punto) Se realizará un programa que reciba los trenes de paquetes y 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), por lo que dicha longitud también debe ser tenida en cuenta a la hora de medir el ancho de banda[1]. 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
3. (1,5 puntos) 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:
   3.1. ¿Qué valores ha empleado para contabilizar las distintas cabeceras?
   3.2. ¿Cuál es la longitud mínima de una de las tramas que se envían? ¿Por qué?
   3.3. ¿Cuál es la longitud máxima de datos que tiene sentido utilizar? ¿Por qué?
   3.4. ¿Con qué longitudes de tren y datos se consiguen mejores resultados? ¿Por qué?

---

1 http://es.wikipedia.org/wiki/Ethernet#Formato_de_la_trama_Ethernet

4. (1 punto) 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. Realice un programa a partir del realizado en el apartado 1. 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.
5. (1,5 puntos) Realice medidas con clienteTren2.py y servidorTren.py, utilizando el emulador que se puede descargar de Moodle. El emulador es un programa que simula retardos variables, pérdidas (al estilo del proxy realizado en la práctica 1) y también anchos de banda. Dicho ejecutable tiene la siguiente sintaxis:
   emulador.exe 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.
6. (1,5 puntos) Capture el tráfico de las medidas realizadas con el emulador y analice con Wireshark el tráfico recibido y compárelo con los resultados obtenidos con su programa.
7. (1,5 puntos) 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 utilice . Igualmente, indique el tamaño del buffer a configurar en el receptor de la llamada para amortiguar el efecto del jitter.
8. (1 punto) 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.

4 Entrega

El trabajo realizado se entregará vía Moodle el día anterior al inicio de la siguiente práctica. 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<4311/4312/4313>-<número de grupo>-P2. Por tanto si una pareja pertenece al grupo 4312 (grupo del martes), y su pareja es la 03, el nombre del fichero de entrega de prácticas deberá ser RM-G4312-03-P2.zip

5 Valoración de las prácticas

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

5.1 Memoria

La memoria debe contener toda la información necesaria para valorar el trabajo realizado en cada uno de los apartados propuestos en esta práctica.

5.2 Criterios funcionales del código presentado

El código presentado contiene la funcionalidad solicitada.

5.3 Evaluación de los conocimientos aprendidos

Los estudiantes serán evaluados tras la entrega respecto de los resultados obtenidos en la práctica mediante un examen.

6 Anexos

6.1 Evaluación del tiempo en Python

La forma en que Python proporciona tiempos es muy dependiente del sistema operativo, variando la precisión del tiempo proporcionado según la función que se utilice para obtenerlo.

De esta manera la función time.time() en Windows, aunque proporciona un valor con 6 decimales, no varía su valor hasta que no pasan varios milisegundos. Por esta razón no se recomienda utilizar esta función.

Como alternativa se plantea la función time.clock(), que tiene una semántica algo distinta, pero que en la implementación en Windows proporciona valores más reales, y permite obtener valores útiles por debajo del milisegundo.

El siguiente código ilustra la diferencia:

```
import time

print ("time.time()")

for i in range(1, 100) :
    print(time.time())

input ("Continuar...")
print ("time.clock()")

for i in range(1, 100) :
    print(time.clock())

input ("Salir...")
```

6.2 Variables globales

En el caso en que sea necesario trabajar con variables compartidas (por ejemplo entre hilos) se puede hacer declarando dicha variable como global en las funciones que la modifican. Si la variable no se modifica no es necesario hacer estas declaraciones. Por ejemplo:

```
globvar = 0

def set_globvar_to_one():
    global globvar #Necesito declararla global para modificarla
    globvar = 1

def print_globvar():
    print(globvar) #No necesito declararla global para leerla

print_globvar() # Imprime 0
set_globvar_to_one()
print_globvar() # Imprime 1
```

6.3 Sockets no bloqueantes:

Para evitar que un socket se quede bloqueado en la función recvfrom() se debe hacer uso de los sockets en modo no bloqueante. Para ello debemos llamar a la función setblocking después de crear el socket y hacer uso de la función select para saber si han llegado mensajes al socket. A continuación se muestra un ejemplo:

```
import socket

import select

s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

s.bind(("127.0.0.1", 9000))

s.setblocking(0)


while True:

    #se indica a select que espere mensajes en el socket s con un

    # timeout de  0.1 segundos

    lect,escr,err=select.select([s],[],[],0.1)

    if len(lect)>0:

        data,addr=s.recvfrom(2048)

        print "Datos recibidos"

    else:

        print "Salta el timeout"
```