DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP3: Capa de Transporte

Teoría de las Comunicaciones

Integrante	LU	Correo electrónico
Furman, Damián	936/11	damian.a.furman@gmail.com
Lambrisca, Santiago	274/10	santiagolambrisca@hotmail.com
Marottoli, Daniela	42/10	dani.marottoli@gmail.com
Vanecek, Juan	169-10	juann.vanecek@hotmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Contents

1 Introducción

La capa de transporte, es la capa de nivel 4 en el modelo OSI, es la capa encargada del transporte de datos de una maquina a otra. Para permitir este intercambio de información y tener la certeza de que la información a sido bien entregada, manteniendo su integridad, existen distintos protocolos. Entre ellos, el mas utilizado es TCP. TCP, nos asegura el envío sin perdida de información, el orden de los datos y la confiabilidad. Para poder cumplir con su objetivo el protocolo debe resolver ciertos problemas que se le presentan en la red, entre ellos nos encontramos con la demora o *delay* y la *perdidadepaquetes*. Estos pueden producirse por diversas causas en la red, y son disparadores de acciones en nuestro protocolo.

El objetivo del trabajo realizado, es poder comprender el comportamiento de la capa de transporte. Para ello, utilizaremos un protocolo de transporte implementado por la cátedra de la materia, sobre el cual efectuaremos ciertos cambios que nos permitan simular los problemas presentes en una red. Pudiendo asi, controlar estas variables, y sacar conclusiones sobre los efectos que estas tienen sobre el comportamiento del protocolo.

2 Desarrollo

Tomando como punto de partida el código suministrado por la cátedra, introdujimos las siguientes modificaciones para agregar dos funcionalidades: por un lado, la introducción de Delay y por el otro, la posibilidad de definir una probabilidad p de pérdida de paquetes con valores que se ubiquen entre el 0 y el 1 (donde 0 no pierde ningún paquete y 1 lo pierde todos). Estos mismos seran efectuados sobre el envio de paquetes de acknowledgment ACK. Provocando que quien envia el paquete, crea que este no fue entregado correctamente.

La implementación se hizo de tal manera que tanto el valor del tiempo de delay como la probabilidad de la pérdida de paquetes se setean como parámetros al ejecutar la función.

Con el fin de agregar estas funcionalidades, se realizaron los siguientes cambios al código original:

- Dentro del archivo **handler.py**:
  - Agregamos la función *se_perdio_paquete* que toma la probabilidad pasada por parámetro y decide sobre la base de esa probabilidad si el paquete que va a enviar efectivamente se va a perder o no.
  - Dentro de la función *send_ack* agregamos un condicional que llama a la función "se_perdio_paquete". Si esta decide que el paquete se perdió, el condicional nos lleva a una sentencia de return y el paquete no se envía. Luego, agregamos también dentro de esta función un Delay simulado mediante la instrucción *time.sleep*() y cuyo valor es un porcentaje, pasado por parámetro, del delay máximo permitido por el protocolo (1 segundo). Para enviar un paquete, la función espera la cantidad de tiempo indicada antes de ejecutar el *build_packet* y el *send*.

Luego de algunas pruebas con estas implementaciones, pudimos observar que aun cuando estabamos perdiendo todos los $ACK's$, el protocolo funcionaba correctamente, asi decidimos que era una buena idea agregar tambien, la pérdida a la actualización de la ventana para realizar nuevas pruebas.

- Dentro del archivo **protocol.py**:
  - Agregamos la función *se_perdio_paquete* que se comporta de manera similar a la mencionada en el item anterior dentro del archivo handler.py
  - Dentro de la función receive se agrega un condicional que llama a la función *se_perdio_paquete* mencionada en el item anterior. El condicional enviará la actualización de ventana unicamente si esta función decide que el paquete a enviar no se va a perder. Si la función retorna que el paquete debe perderse, no se envía la actualización de ventana y se imprime por consola que el paquete se perdió.

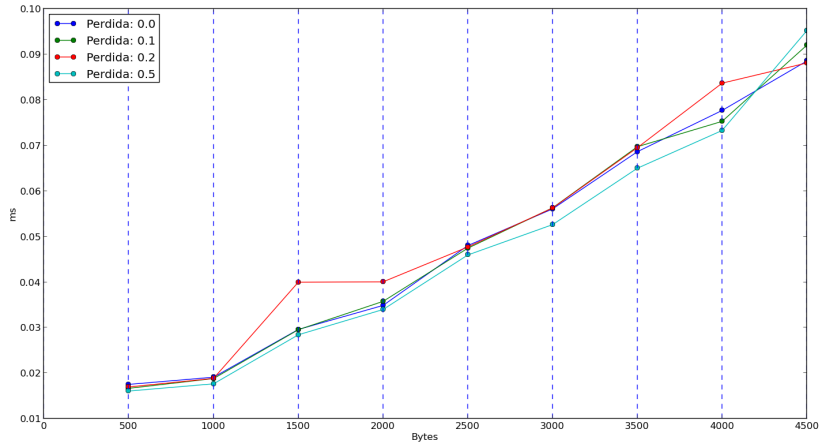

Figure 1: caption

Tiene el max buffer seteado en 500 Bytes. Se evaluó el tiempo que tarda el protocolo en función del tamaños, y distintos porcentajes de perdidas. El delay esta seteado en 0.

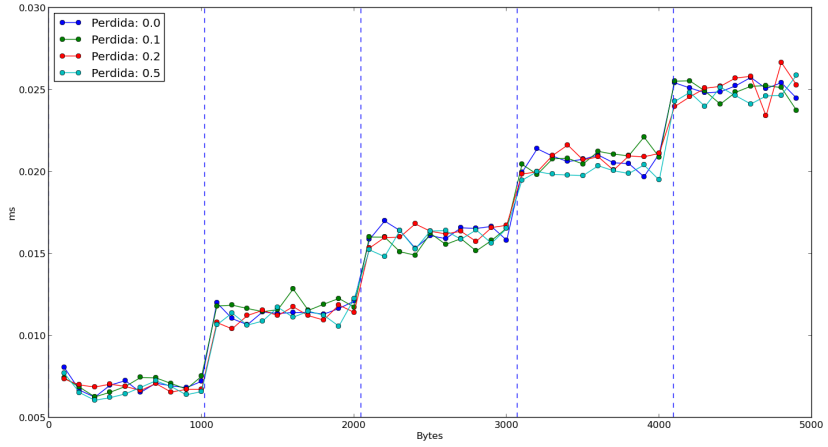

Figure 2: caption

El mismo que el anterior pero con un buffer de 1024, y una mayor granularidad en los tamaños.

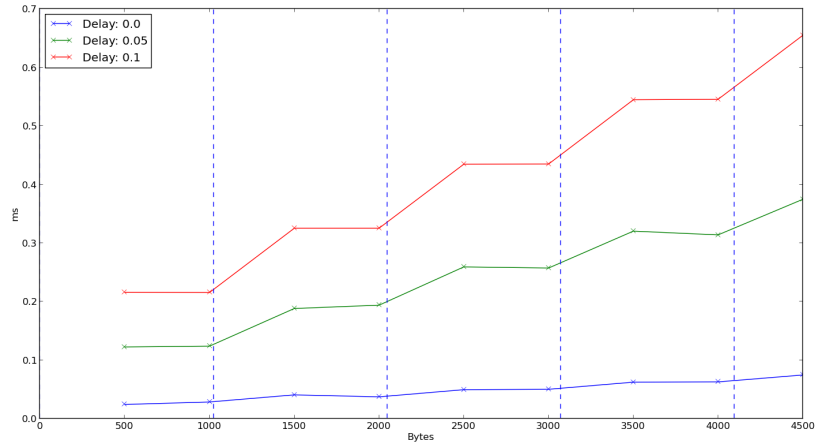

Figure 3: caption

Tiene el max buffer seteado en 1024 Bytes. Se evaluaó el tiempo que tarda el protocolo en función del tamaños, y distintos porcentajes de delay. La perdida esta fijada en 0.

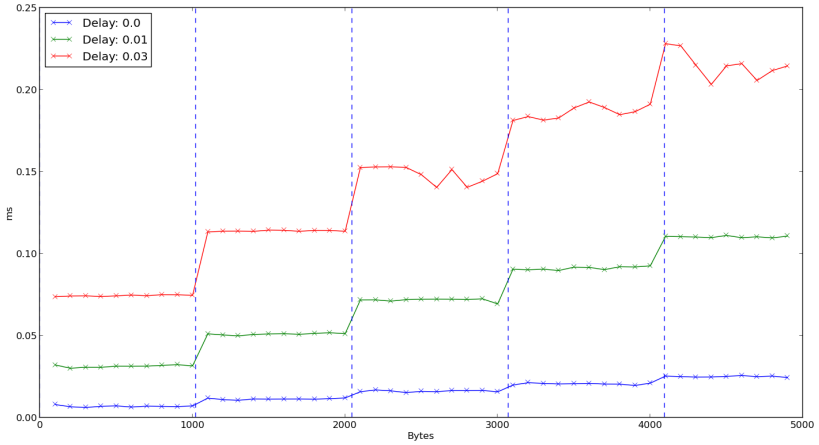

Figure 4: caption

El mismo que el anterior pero más granularidad en los tamaños evaluados.

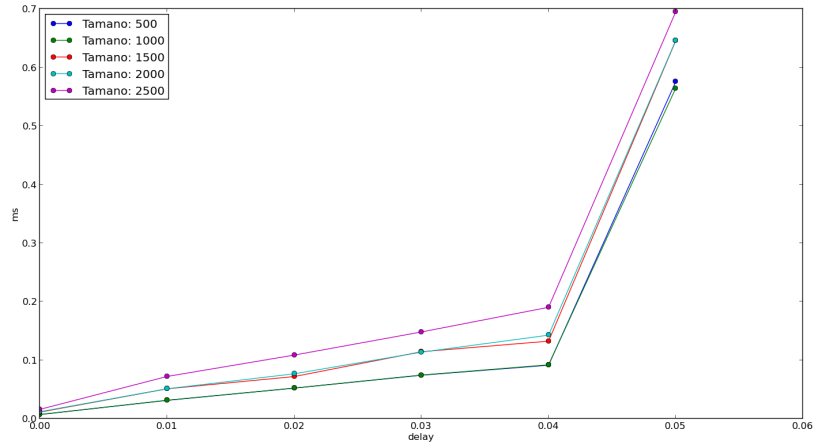

Figure 5: caption

Buffer 1024 bytes.

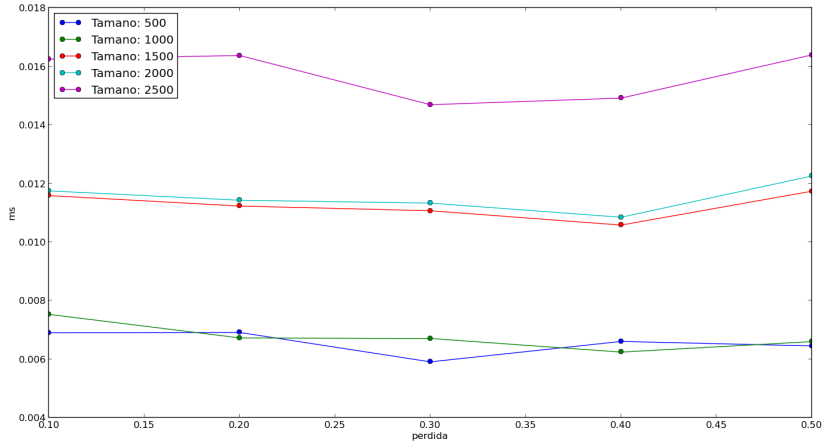

Figure 6: caption

Buffer 1024 bytes.