Programación de Sistemas

Tarea 4: Emunador de Variaciones de Retardo y Pérdidas en Transferencias UDP

Integrantes:	Felipe Acevedo Niklas Tampier
Profesor:	Agustín González
Fecha:	28-11-2014

PROBLEMA

En ocasiones nos vemos enfrentados a la necesidad de correr una aplicación que establece conexiones a través de Internet y nos interesaría averiguar cuál sería su comportamiento si las transferencias extremo a extremo poseen retardo y pérdidas de paquetes mayores a las de una red local. Por ejemplo, cuando deseamos controlar un robot remoto a través de Internet. Generalmente los desarrollos se hacen en un laboratorio donde la red de área local ofrece 100Mbps de tasas en nivel físico. Surge la pregunta ¿Cómo podemos correr un proyecto haciéndolo creer que tiene mayor retardo y mayor tasa de pérdida de paquetes?.

SOLUCIÓN

Implementar un programa que funcione como intermediario entre un cliente y un servidor. Su función es simular un retraso en los paquetes UDP como también las posibles perdidas.

IMPLEMENTACIÓN

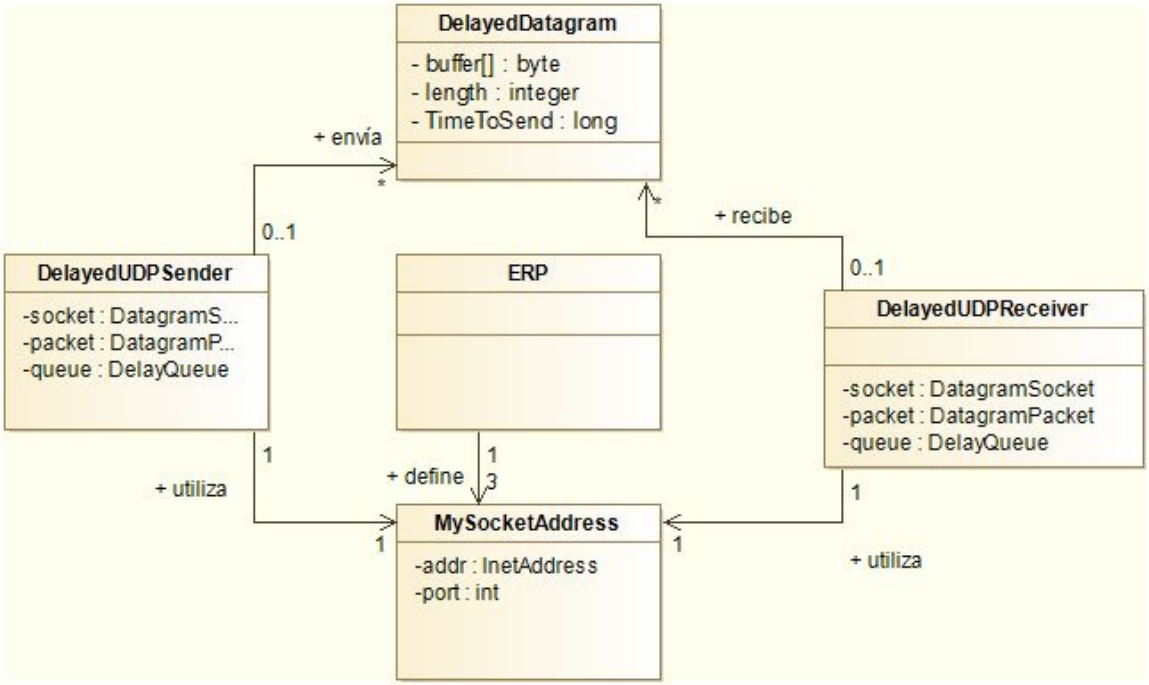

Figure 1: Diagrama de Dominio

Principales clases utilizadas:

- **DelayedUDPSender:** Esta clase es la encargada de enviar los paquetes tanto al servidor como al cliente, los paquetes enviados al servidor son los que provienen del cliento y viceversa.

- **DelayedUDPReceiver:** Esta clase cumple dos funciones, la primera es recibir los paquetes que son enviados al servidor desde el cliente y los paquetes que son enviados al cliente por parte del servidor. La segunda función de esta clase es simular la pérdida y retraso de los paquetes según los datos ingresados.

- **ERP:** Clase principal encargada de inicializar las 4 hebras necesarias para establecer la conexión entre el servidor y el cliente, además de recibir los parámetros ingresados por el usuario al momento de la ejecución de este simulador de retrasos y pérdida de paquetes. Estos parámetros son entregados a las hebras, dos para enviar y dos para recibir.

- **DelayedDatagram:** Esta clase simula los paquetes con todos los parámetros necesarios incluyendo el delay de estos, además implementa la idea de hacer una cola FIFO, para evitar así el solapamiento del envío de paquetes.

- **MySocketAddress:** Con esta esta clase se manejan los datos tanto del servidor como el cliente (puertos de escucha y direcciones) para poder realizar los envios correspondientes.