LABORATORIO 3

TRANSFERENCIA DE DATOS CONFIABLE RDT1.2

Propósito

Aprender a usar la API de sockets provista por la python. Usar clases para haces uso del objeto Paquete y de la red Network provistos por la cátedra.

Generalidades

Escriba y testee un programa `emisor_rdt1.2` y un programa `receptor_rdt1.2` en una misma computadora. Luego ejecute ambos programas en computadoras separadas, y verifique que se pueden comunicar. Para ello, consideramos el protocolo analizado en clases `rdt1.0`.

Para este proyecto, deberá modificar el Laboratorio 2, de manera que ahora permita crear, enviar y recibir **paquetes**, una nueva clase de objetos provista por la cátedra, en el archivo `paquete.py` pero a través de una nueva **red**. Para ello, ambos programas le han sido provistos con poca funcionalidad implementada. Aquello que le será necesario para que todo funcione correctamente. El resto de las funciones con la leyenda `# IMPLEMENTAR`, deberán ser escritas por usted.

A su vez, deberá hacer uso de constantes de configuración que provee el archivo `constantes.py`. En la misma se presentan valores que vamos a usar tanto en el emisor como en el receptor. La idea de este archivo, es que nos permita definir dichos valores en un sólo lugar y luego poder usarlos en otros lugares mediante sus variables.

Procedimientos y Detalles

1. Investigue la API de sockets provista por el lenguaje de programación Python. Para ello deberá poner especial énfasis en los sockets que hacen uso del protocolo UDP. El capítulo 2.7 provisto por la cátedra es la lectura sugerida al respecto.

2. Desarrolle un programa `emisor_rdt1.2.py` modificando `emisor_rdt1.1.py` para que ahora, no envíe directamente el mensaje al receptor, sino que lo haga a la `RED`. **Observaión:** La red posee su dirección IP y su dirección de puerto. Son las contantes: `NETWORK_IP, NETWORK_PORT`. Deberá hacer uso de las mismas, cuando envíe un paquete desde el lado emisor.

3. Desarrolle un programa `receptor_rdt1.2.py` que deberá aguardar infinitamente por mensajes de la capa de red.

4. Realizar un test de loopback con el `emisor_rdt1.2.py` y el `receptor_rdt1.2.py` ejecutándose en una misma computadora como se explicó en clases.

5. Testear el cliente y el servidor ejecutando el primero en una computadora y el segundo en otra.

6. Modificar el servidor para que no sólo imprima el mensaje en pantalla, sino que además guarde un registro de todas las palabras recibidas y que haya impreso.

7. Como los puertos de conexión son de carácter efímeros, vamos a requerir pasar por parámetro dicho valor. Como también el servidor se va a ejecutar en diferentes máquinas, el cliente debe de poder recibir no sólo el puerto como parámetro, sino también la dirección ip del servidor con quien intenta establecer comunicación.