LABORATORIO 1

TRANSFERENCIA DE DATOS CONFIABLE RDT1.0

Propósito

Aprender a usar la API de sockets provista por Python.

Generalidades

Modifique y testee un programa cliente `rdt1.0` y un programa servidor `rdt1.0` 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 ello, han sido provistos por la cátedra un par de programas python, `UDPserver.py` y `UDPclient.py`, para que pueda basar su programa. Los mismos han sido desarrollados siguiendo como guia a los presentados en el capítulo 2.7 del teórico.

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 `servidor_rdt1.0.py` que deberá aguardar infinitamente por mensajes de los clientes. El mismo deberá implementar las funcionies:

   a. `socket = create_socket(adrss, port)`: Crea un socket con las direcciones provistas como parámetros.

   b. `packet = rdt_rcv(socket)`: permite recibir un paquete a traves de un socket UDP.

   c. `extract(packet, data)`, tal que extrae del paquete la información contenida en el mismo;

   d. `deliver_data(data)`, que recibe datos y, por el momento, simplemente los imprime por pantalla.

   e. `close_socket(socket)`: Se encarga de cerrar el socket creado.

3. Desarrolle un programa `cliente_rdt1.0` que deberá aguardar infinitamente por mensajes que la capa de aplicacion leerá desde consola para luego enviarlos a traves de un puerto UDP al servidor. Para ello, debería implementar las siguientes funcionalidades:

   a. `socket = create_socket(adrss, port)`: Crea un socket con las direcciones provistas como parámetros.

   b. `rdt_send(socket, data)`: Envía los datos leídos desde la capa de aplicación a la capa de transporte

   c. `packet = make_pkt(data)`: crea el paquete en la capa de transporte. Para este primer laboratorio, el paquete van a ser los mismos datos. A medida que los laboratorios evolucionen, van a presentar un formato diferente

   d. `udp_send(packet)`: Envía el paquete a traves de UDP, al servidor.

   e. `close_socket(socket)`: Cierra el socket pasado como parámetro.

4. Para testear el `servidor_rdt1.0.py`, vamos a necesitar un único *número de aplicación*. Si múltiples grupos están usando el laboratorio, será necesario para cada servidor corriendo que se le sea asignado un único número. Ya sea que su profesor le asigne un número único o coordinando entre ustedes para elegir los siguientes valores comenzando en 2001, 2002, 2003, y así sucesivamente. Anotar el número abajo.

   Número asignado:

6. Realizar un test de loopback con el `servidor_rdt1.0.py` y el `cliente_rdt1.0.py` ejecutándose en una misma computadora como se explicó en clases. Use el nombre de computadora `localhost` y el número de aplicación que se le fue asignado en el paso anterior.

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

8. 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.

9. 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.