LABORATORIO 1

MINI APLICACIÓN HTTP

Propósito

Aprender a usar la API de sockets provista por Python.

Generalidades

Modifique y testee un programa cliente `client.py` y un programa servidor `server.py` 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 `http`.

Para ello, han sido provistos por la cátedra un par de programas python, `TCPServer.py` y `TCPClient.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 TCP. El capítulo 2.7 provisto por la cátedra es la lectura sugerida al respecto.

2. Desarrolle un programa `server.py` que deberá aguardar infinitamente por mensajes de los clientes. El mismo deberá implementar las funciones:

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

   b. `data = receive_data(socket)`: permite recibir un paquete a traves de un socket TCP y me devuelve los datos que transporta el mismo.

   c. `send_data(data)`, recibe datos y, los envía por el socket a su destinatario

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

3. Desarrolle un programa `client.py` que deberá aguardar infinitamente por mensajes que la capa de aplicacion leerá desde consola para luego enviarlos a traves de un puerto TCP 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. `send_data(socket, data)`: Envía los datos leídos desde la capa de aplicación a la capa de transporte

   d. `data = receive_data(socket)`: Recibe los datos desde el socket.

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

4. Desarrolle un protocolo de intercambio de información entre sus dos programas anteriores. El mismo debe de cumplir con las especificaciones establecidas a continuación.

   a. `LIST`: Este comando me permite listar los objetos presentes en el directorio raíz donde se ejecuta el servidor. La respuesta deberá presentar como encabezado, el resultado de la operación seguido de tantas líneas como archivos presentes existan.

   b. `GET <FILE>`: Me devuelve un archivo presente en el directorio raíz donde se ejecuta el servidor. La respuesta deberá presenta como encabezado, el resultado de la operación segudido del archivo mismo.

   c. `METADA <FILE>`: Me devuelve la metada de un archivo presente en el directorio raíz donde se ejectua el servidor. La respuesta deberá presentar como encabezado el resultado de la operación seguido de los metadatos del mismo.
   d. `CLOSE`: Cierra la conexión entre ambas partes. La respuesta deberá presentar como encabezado el resultado de la operación y luego terminar la conexión.

5. Para testear el `server.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 `server.py` y el `client.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. Modificar el servidor para que no sólo imprima el mensaje en pantalla, sino que además guarde un registro de todas las comandos recibidos junto con su solicitante.

8. 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.