Tarea 3

Sistemas Operativos y Redes

Universidad de los Andes

-

Integrantes:

Francisco Pieper (GitHub: panchoapc)

Nicolas Apara (GitHub: nccaster)

1. Formato

Nuestro protocolo será similar a HTTP porque nos interesa saber los métodos de comunicación entre el cliente y el servidor. Estos serán:

a. GET:
   Con la cual el cliente podrá pedir recursos del servidor, como también el servidor podrá reconocer dicha acción.
   Como el servidor maneja una base de datos del tipo Key-value, las peticiones de recursos de cliente seran segun key o segun value.

b. PUT:
   El cliente será capaz de realizar un upload de un recurso por medio de una conexión socket establecida con el servidor.

c. DELETE: Para borrar lo especificado por el cliente, ya sea utilizando como referencia el 'value' o el 'key' del elemento en cuestión.

d. UPDATE:
   El cliente será capaz de actualizar algún recurso de la base de datos

e. CONNECT: Para verificar si la conexión entre host y cliente se ha establecido.
f. DISCONNECT:
   En caso en que el cliente cierre la conexión o bien cuando el servidor cierre la conexión(si es que limitamos el tiempo de conexión para cada cliente).
   Una vez se pida disconnect se deberá cerrar el socket como también el thread con el cual se ha conectado el cliente

También se utilizara una comunicación tipo TCP para especificar el puerto de destino, los métodos del mensaje y el puerto de origen.

Para ejemplificar, un formato del mensaje para el método GET seria:

<host_port> <client_ip> <algun_valor_de_validacion> GET key?value

En nuestro protocolo, luego de cada petición del cliente, existirá una respuesta por parte del servidor acusando recibo. Después, el servidor manda el resultado y es ahora el cliente que debe responder si se recibió. En caso de no recibir respuestas tras un tiempo determinado, se debe repetir el envío.

Se deberá mantener la conexión abierta desde servidor al cliente hasta que se indique cierre a través del comando 'disconnect'.

Para mantener la conexión buscaremos guardar los estados del cliente y que estos sean temporales, por ejemplo, que dure 5 minutos ya que todas las peticiones son cortas y, de este modo, poder ir liberando recursos en el servidor para poder aceptar nuevos clientes.

En el caso en que se pierda la conexión con el cliente, se deberá cerrar el socket (si existe alguno) y terminar el thread correspondiente. Por lo tanto, el cliente deberá hacer una petición nueva de conexión.

En el caso de que el cliente quiera conectarse al servidor pero el servidor no esté activo o falle la conexión, se deberá informar sobre lo sucedido (al cliente).

2. Interacción

La interacción será no orientado a la conexión pero con acuso de recibo, ya que se espera un tipo solicitud y respuesta para acceder y/o modificar la base de datos que maneja el servidor. Esta interacción, al ser cliente/servidor deberá contar con diversos requisitos.

Para el caso del servidor, este deberá:

- Estar disponible y funcionando
- Tener una dirección ip permanente
- Ser multithreaded para tener disponible una conexión a varios clientes de ip distintas.

Para el caso de los clientes, estos deberán:

- Comunicarse con el servidor
- Avisar en caso de no poder conectarse, ya sea por falla en el servidor o por falla en el cliente
- Iniciar todas las peticiones ya sea de conexión, pedir datos, actualizar o eliminar.

3. Diagrama de interacciones