Julio Castro
Richard Le May
05-11-19

Informe tarea 3, Parte 1:

Antes de definir el formato de nuestros mensaje, se dara una explicacion de como funciona la comunicación entre cliente servidor.

A grandes rasgos, se utilizara el protocolo HTTP para la transmisión de información mediante la internet entre el cliente y el servidor. Por un lado, el cliente se encontrara corriendo en nuestro navegador de preferencia, mientras que el servidor se encontrara corriendo sobre un dominio HTTP específico, dado por el host y el archivo que se comparte(de haber uno).

Por ejemplo, si el servidor se encuentra en el puerto 8080 de nuestro localhost y se desea compartir cierto documento ‘doc.html’, el link del dominio será: ‘localhost:8080/doc.html’.

Con todo esto en mente, si nos conectamos al servidor mediante nuestro cliente, deberíamos poder empezar a ejecutar las funcionalidades de nuestro programa original mediante el navegador.

Mensajes del protocolo:

Definiremos un formato estandarizado basado en la estructura petición/respuesta HTTP. Se tendrán dos tipos de mensajes generales, uno será una petición, la cual será enviada por el cliente y el otro será una respuesta, la cual será enviada por el servidor al cliente.

Ambos formatos son muy parecidos entre si, solo difieren en el contenido que poseen.

Petición:

Cabeza
Operación
Data

En el caso de la petición, se tienen 3 casillas de información:

- **Cabeza**: Contiene información variada del host, servidor y cliente, como las direcciones de cada uno, navegador usado por el cliente, etc.

- **Operación**: Hay 7 tipos de operación:
    - Insert(k,v)
    - Insert()
    - Get()
    - Peek()
    - Update()
    - Delete()
    - List
- **Data**: Dependiendo de la operación dada se recibirá uno o dos valores, siendo estos key y/o value. El formato de entrega dependerá de los 2 diferentes casos:
    - Sólo hay un valor: valor
    - Hay dos valores: key-value
    - Ejemplos:
        - Si queremos utilizar insert(2,”hola”):

Host: localhost:8080 Navegador: Chrome . .
Insert(k,v)
2-”hola”

        - Si queremos utilizar get(2):

Host: localhost:8080 Navegador: Chrome . .
Get()
2

Respuesta:

Estado
Cabeza
cuerpo

En el caso de la respuesta, se tienen 3 casillas de información:

- **Estado**: Esta casilla contiene un código de estado HTTP, el cual indica el resultado de la operación solicitada por el cliente. Se basa en el estándar para códigos de estado, el cual posee los siguientes formatos de códigos:
    - **2XX**: Peticiones correctas: Se entregan cuando la petición hecha por el cliente fue procesada correctamente. en nuestro programa se contarán con los siguientes códigos:
        - 200: Respuesta estándar para peticiones correctas.
        - 210: Insert(key,value) exitoso.
        - 211: Insert(value) exitoso.
        - 220: Get(key) exitoso.
        - 230: Peek(key) exitoso
        - 240: Update(key,value) exitoso.
        - 250: Delete(key) exitoso.
        - 260: List exitoso.
    - **4XX**: Errores: Se entrega cuando la petición hecha por el cliente falla. En nuestro programa, se cuenta con los siguientes códigos de error:
        - 400: Error genérico.
        - 410: Error al insertar.
        - 420: Error al encontrar key(key inexistente)
        - 430: Key invalida(valor no procesable, ej: caracteres no numéricos).
        - 450: Error de conexión.

- **Cabeza**: Contiene información variada del host, servidor y cliente, como las direcciones de cada uno, navegador usado por el cliente, etc.

- **Cuerpo**: Contiene la información entregada por el servidor al cliente. el formato de entrega será el siguiente:

INFO: val1-val2-...-valn

    En donde cada valor ‘n’ entregado por el servidor, es puesto en orden en un buffer.

Ej:
Si tenemos que el cliente pide la lista de las claves mediante el comando ‘list’, se generará la siguiente respuesta:

260
Host: localhost:8080 Navegador: Chrome . . .
INFO: 506-507-508-509-1990-1991-5555-9455

Esto le dirá al cliente que la operación fue exitosa, y las claves existentes son: 506,507,508,509,1990,1991,5555 y 9455.

Interacciones posibles:

En nuestro programa, las interacciones posibles entre cliente servidor son bien simples y prácticamente lineales:

- Para empezar, un cliente se conecta en el servidor.
- Luego el cliente puede hacer múltiples peticiones dentro de una sesión, siempre que la sesión siga conectada.
- Cuando ya no se desee hacer más peticiones, el cliente se desconecta del servidor.

Esto se puede graficar mediante el siguiente diagrama de flujo: