



Universidad de
los Andes



**FACULTAD
DE INGENIERÍA
Y CIENCIAS
APLICADAS**

Sistemas Operativos y Redes

Tarea tres

Informe de protocolo mejorado

Integrantes:

- > Ignacio F. Garcés Santander
- > Francisco J. Jiménez Iglesias

1. CAMBIOS CON RESPECTO AL INFORME DE LA PARTE 1

- Sí, hubo cambios, para mejorar el protocolo.
- ValType: Se agregó la posibilidad de definir el tipo de dato (value) que puede tener una llave (key) asociada.
- La estructura del *request* del cliente cambió, ya que, como es TCP, requiere de una inicialización de conexión, y mantenerla a lo largo de la *conversación* cliente-servidor, así que sólo es necesario ingresar puerto y dir. IP al conectar. En dicho caso, en el comando connect, ya no se requiere el puerto del cliente, pues el *request* se hace desde el cliente y al conectarse en el server se crea un socket nuevo con un puerto dado por el sistema operativo.
- Algunos nombres asociados a códigos de retorno fueron cambiados.

2. PROTOCOLO: INFORMACIÓN GENERAL

Es un protocolo con política de comunicación confiable, con TCP.

3. REQUESTS (CLIENTE)

Forma (basada en HTML):

```
<command>  
[other_parms]
```

command puede ser:

- connect: se establece una conexión entre el cliente y el servidor.
- disconnect: se cierra conexión.
- quit: cierra el programa.
- insert: inserta un valor en la BD, especificando llave o no.
- get: obtiene el valor de una llave.
- peek: verifica si existe una llave en la base de datos.
- update: cambia el valor asociado a una llave.
- delete: borra una llave de la BD.
- list: solicita una lista de todos

other_parms son los argumentos de las funciones en la tarea 1. Es decir, sólo se usa en los comandos insert, get, peek, update, delete. Ver tabla 3.1. En ella, key y value son datos tipo int, y <null> significa que no existe other_parms, es decir, es un *string* vacío. Se puede especificar el tipo de dato que se puede insertar asociado a una llave, en el comando insert, usando el parámetro Valtype como muestra la tabla 3.1. Si dicho parámetro no se especifica, por defecto se toma como *string* (str). Para todo comando, los parámetros deben darse en orden. En connect, no se incluye el puerto de cliente porque el sistema operativo lo asigna al conectar el socket.

command	other_parms
connect	Server IP: <host IP> Server port: <server port>
disconnect	<null>
quit	<null>
insert	[Key: <key>] Value: <value> [ValType: bin int float str]
get	Key: <key>
peek	Key: <key>
update	Key: <key> Value: <value> [ValType: bin int float str]
delete	Key: <key>
list	<null>

Tabla 3.1: comandos y sus parámetros extra asociados.

4. RESPONSES (SERVIDOR)

```
<mensaje_de_estado> (code: <código_de_estado>)
Body: <mensaje_servidor>
```

Por defecto, mensaje_servidor es vacío. Sólo si la respuesta debe tener un mensaje (como el comando list del cliente), entonces su mensaje es distinto de vacío.

En las tablas 4.1 y 4.2, la columna "Código" es el valor de código_de_estado y "Mensaje de estado" es mensaje_de_estado.

4.1. ESTADOS DE RESPUESTA

Código	Mensaje de estado	Descripción
0	Connection successful	Conexión establecida satisfactoriamente entre el servidor y el cliente.
1	Insert successful	La insercion de valor es realizada exitosamente, retorna un int con la clave
2	Get successful	La obtencion de valor ha sido exitosa, retorna int con el valor asociado a la key enviada
3	Peek successful	La operación peek ha sido completada, retorna un bool si esta o no en la base de datos
4	List successful	Accedió y retorna (en items) al cliente la lista de pares clave-valor (llave-valor).
5	Update successful	Logró sobrescribir el valor asociado a una llave.
6	Delete successful	Par clave-valor eliminado satisfactoriamente.

Tabla 4.1: respuestas de éxito.

Código	Mensaje de estado	Descripción
100	Connection timeout	El tiempo para esperar a que el cliente se conecte expiró (el tiempo límite que decía la tarea 1).
101	Server timeout	Se agotó el tiempo de respuesta del servidor (digamos, 5 segundos).
102	Connection failed	No se ha podido realizar una conexión con el servidor.
103	Evil access	Intento de acceder a la BD sin conectarse primero usando el comando connect. Intento de acceso malvado.
104	Evil command	command del cliente no reconocido.
105	Evil key	Se trató de usar un parámetro key inválido en algún comando que lo requiera: la llave debe ser un int no negativo.
106	Bad key read	Se intentó leer una llave no existente en la BD.
107	Bad key write	Se intentó eliminar o editar una llave no existente en la BD.
108	Bad key overload	Se intentó insertar un valor con llave ya existente. (Debió haber usado comando update)
200	Bad request syntax	La petición del cliente no está escrita en forma correcta.
201	Bad parameters	Comando reconocido, pero other tags entregados no son válidos para dicho comando.
202	Bad host	Host inválido.
203	Bad client port	Error al conectar usando el puerto de cliente especificado.
204	Bad server port	Error al conectar usando el puerto de servidor especificado.
205	Evil parameter	Valor entregado de los parámetros no pudo ser convertible al tipo esperado. Ej.: <i>string</i> no convertible a <i>int</i> .

Tabla 4.2: respuestas de excepción.

5. EJEMPLOS

5.1. EJEMPLO 1

Lo que en la tarea 1 era:

REQUEST
peek(89)

RESPONSE
false

Con el protocolo definido, queda:

REQUEST
> peek ... Key: 89 ...

RESPONSE
Received from server: 'Peek sucessfull (code: 3) Body: false'

5.2. EJEMPLO 2

REQUEST
> connect ... Server IP: 127.0.0.8 ... Server port: 7001 ...

RESPONSE
Received from server: 'Connection successful (code: 0)'

5.3. EJEMPLO 3

REQUEST
> list ...

RESPONSE
Received from server: 'List successful (code: 4) Body: Key Value ValType 1 -4 int 5 309.5 float 2 "HH" str 7 134587 int'

5.4. EJEMPLO CUATRO

REQUEST
> insert ... key = 3 ... value = 9999

RESPONSE
Received from server: 'Insert successful (code: 2) Body:'

5.5. EJEMPLO CINCO

REQUEST
> qe pasa ... 2335476: -1

RESPONSE
Received from server: 'Evil command (code: 104) Body:'

(este sólo si el cliente se encuentra conectado al servidor actualmente)

6. DIAGRAMA

El diagrama de comunicación de ejemplo se ve en la firugra 6.1.

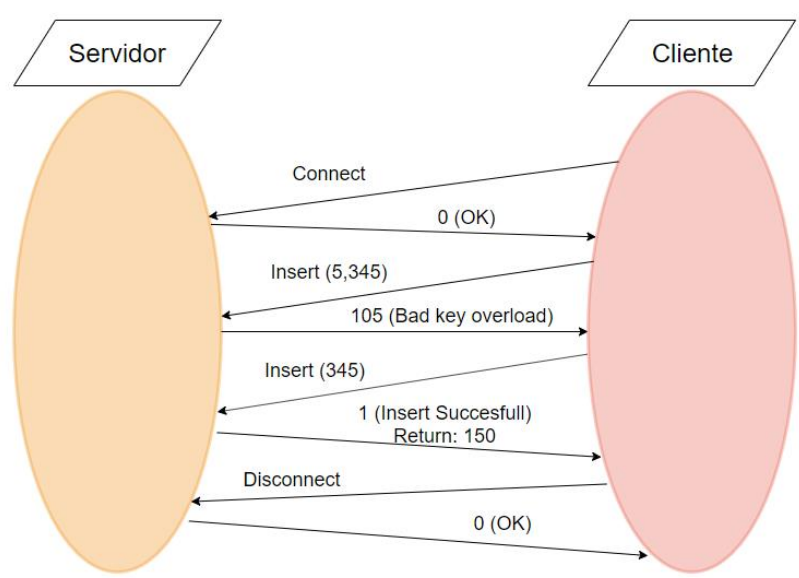


Figura 6.1: diagrama

7. REFERENCIAS

- An overview of HTTP. (2019). Retrieved 1 November 2019, from <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>
- Anexo: Códigos de estado HTTP. (2019). Retrieved 5 November 2019, from https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP