Redes de Computadoras

Obligatorio 2 - 2022

Facultad de Ingeniería
Instituto de Computación
Departamento de Arquitectura de Sistemas

Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en el EVA.

En particular está prohibido utilizar documentación de otros estudiantes, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (EVA, news, correo, papeles sobre la mesa, etc.).

Introducción

Forma de entrega

Una clara, concisa y descriptiva documentación es clave para comprender el trabajo realizado. La entrega de la tarea consiste en un único archivo obligatorio2GrupoGG.tar.gz que deberá contener los siguientes archivos:

- Un documento llamado Obligatorio2GrupoGG.pdf donde se documente todo lo solicitado en la tarea. GG es el número del grupo. La documentación deberá describir claramente su solución, las decisiones tomadas, los problemas encontrados y posibles mejoras.
- El código fuente del programa (**en lenguaje Python**) e instrucciones claras de cómo ejecutar el sistema.

La entrega se realizará en el sitio del curso, en la plataforma EVA.

Fecha de entrega

Los trabajos deberán ser entregados **antes del 16/10/2022 a las 23:30 horas**. No se aceptará ningún trabajo pasada la citada fecha y hora. En particular, no se aceptarán trabajos enviados por e-mail a los docentes del curso.

Objetivo del Trabajo

Aplicar los conceptos teóricos de capas de aplicación y transporte, la utilización de la API de sockets TCP y UDP, y la arquitectura de aplicaciones cliente-servidor y Peer-to-Peer [1].

Descripción general del problema

Se desea implementar una Base de Datos *clave-valor*[1] distribuida.

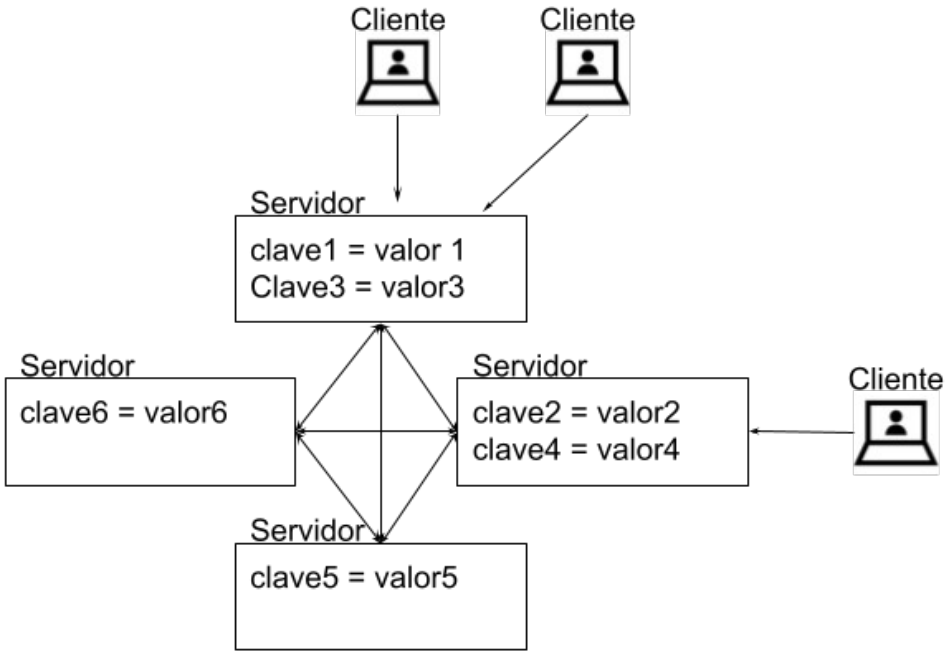
[1] https://es.wikipedia.org/wiki/Base_de_datos_clave-valor

Problema a resolver

Se propone desarrollar una base de datos de claves-valor. Las claves y los valores son cadenas alfanuméricas sin espacios en blanco.

El sistema está formado por aplicaciones servidor y cliente. El servidor es el responsable de almacenar los datos y responder a solicitudes de los clientes. El cliente provee una interfaz al usuario y realiza solicitudes al servidor.

Los datos pueden ser almacenados de forma distribuida entre varias instancias de la aplicación servidor. La base de datos es fácilmente escalable agregando nuevas instancias del servidor. Los datos son almacenados en alguno de los servidores disponibles, de forma transparente para los clientes finales.



1. Servidor

El servidor es el componente principal del sistema. Implementa las siguientes funcionalidades:

- Es capaz de descubrir otras instancias del servidor en una red local por el mecanismo descrito en 3.1 (DESCUBRIMIENTO).
- Es capaz de gestionar datos controlado por clientes que se comunican mediante el protocolo 3.2 (DATOS).
- Es capaz de comunicarse con otras instancias del servidor usando un protocolo especificado en 3.2 (DATOS).
- Es capaz de distribuir la información entre los servidores presentes. Para esto, es capaz de determinar el servidor designado para almacenar ese dato, y en caso que se le solicite manipular un dato perteneciente a otro servidor se encargará de realizar las operaciones necesarias.
- Es capaz de almacenar y recuperar información en formato clave-valor desde memoria RAM.

Se sugiere la siguiente forma de invocación:

```
$ python server.py <ServerIP> <ServerPort>
```

donde:

- <ServerIP>
  Dirección IP donde el servidor aceptará conexiones, por ejemplo 127.0.0.1
- <ServerPort>
  Puerto donde el servidor aceptará conexiones, por ejemplo 2022

1.1 Distribución de datos

Toda clave tiene un único servidor asignado responsable de almacenarlo. Cuando un servidor recibe una petición de operar sobre una clave, deberá determinar el servidor responsable y realizar la operación correspondiente sobre el. Nótese que un cliente puede estar conectado a un servidor cualquiera, por lo tanto, el servidor deberá determinar cada vez al servidor responsable de esa clave (que eventualmente podría ser el mismo). Una vez determinado el servidor responsable, la operación se debe impactar en el.

Para determinar el servidor responsable de una clave se realiza el siguiente procedimiento:

- Todo servidor tiene asignado una firma en la forma de un número de 32 bits, resultado de aplicar la función CRC32 sobre el string "<*ServerIP*>:<*ServerPort*>". Por ejemplo, al servidor de arriba con string "*127.0.0.1:2022*" le corresponde la firma 0x727BC764.
- A cada clave se le asigna una firma de la misma forma. Por ejemplo, a la clave "clave1" le corresponde la firma 0xDA9600FD.
- A una clave le corresponde el servidor cuya firma es el número más cercano a su propia firma. En case de empate se elige el que tiene la firma más baja. Por ejemplo, si además del servidor anterior hubiera uno en la misma IP pero puerto 2023 (por la tanto firma 0x6B60F625), el correspondiente a la clave "clave1" sería el primero.

Se podrá ver un ejemplo de ejecución del sistema en el Anexo 1.

2. Cliente

Un cliente es capaz de conectarse a un servidor que se le indique, y realizar una operación de alta, baja o consulta de un valor. La comunicación con el servidor se realiza usando el protocolo 3.2 (DATOS).

Se sugiere la siguiente forma de invocación:

```
$ python cliente.py <ServerIP> <ServerPort> <Op> <Key> [<Value>]
```

donde:

- <ServerIP>
  Dirección IP del servidor al que se desea conectar, por ejemplo 127.0.0.1
- <ServerPort>
  Puerto del servidor al que se desea conectar, por ejemplo 2022
- <Op>
  Operación a realizar: GET (leer un valor e imprimirlo), SET (escribir un valor) o DEL (borrar un valor).

- <Key>
  Clave del valor que se desea leer, escribir o borrar.
- <Value>
  Puede ser un argumento del comando cuando <Op> es SET para indicar el valor a almacenar. Si se omite en la operación SET, el valor a escribir se lee de la entrada estándar.

3. Protocolos

Se definen dos protocolos. El primero (DESCUBRIMIENTO) es usado para que los servidores se descubran entre ellos. El segundo (DATOS) es usado por los servidores y los clientes para administrar los datos almacenados.

3.1 Protocolo: DESCUBRIMIENTO.

Este protocolo es implementado por los servidores. Está basado en mensajes UDP de broadcast. Por defecto se usará el puerto 2022. El protocolo asume que todos los servidores están conectados a la misma red local.

El protocolo consiste de un único mensaje:

3.1.1 ANNOUNCE

Este mensaje se emite periódicamente cada 30 segundos mientras el servidor está activo. El mensaje es de texto plano, y tiene el siguiente formato:

```
ANNOUNCE <port>\n
```

El campo <port> es el puerto TCP donde el servidor acepta conexiones para el protocolo DATOS. Se asume que el servidor escucha en la dirección IP desde donde fue emitido el mensaje UDP ANNOUNCE.
Un servidor está continuamente escuchando por mensajes de este tipo, y cuando detecta la presencia de un nuevo servidor establece una conexión TCP al servidor detectado, para soportar el protocolo DATOS.
Cada vez que se detecta un nuevo servidor, cada servidor deberá recalcular la asignación de las claves que mantiene almacenadas, y entregar las claves que corresponda a su nuevo responsable.

3.2 Protocolo: DATOS

Este protocolo se usa para realizar las operaciones con los datos. Es orientada a texto, con mensajes separados por caracteres de linea nueva (\n). Las comunicaciones tienen la forma pregunta/respuesta, donde el intercambio es siempre iniciado por el nodo que abrió la conexión.

Los mensajes son:

3.2.1 GET

```
GET <key>\n
```

Se consulta por el valor asociado a la clave *<key>.* Como respuesta se puede generar uno de dos mensajes posibles:

`OK <value>`**`\n`**
para devolver el valor, o

`NO`**`\n`**
en caso de no existir esa clave en la base de datos.

3.2.2 SET

`SET <key> <value>`**`\n`**

Le asigna a la clave *<key>* el valor *<value>.* Si la clave ya existe, se modifica el valor asociado. Como respuesta se genera el siguiente mensaje:

`OK`**`\n`**

3.2.3 DEL

`DEL <key>`**`\n`**

Elimina la entrada con clave *<key>* de la base de datos. Como respuesta se genera el siguiente mensaje:

`OK`**`\n`**

4. Se pide

a) Diseñe y documente la arquitectura de la aplicación descrita anteriormente.

b) Implemente en lenguaje Python, utilizando las primitivas de la API de sockets correspondiente [4][5][6], las aplicaciones cliente y servidor descritas anteriormente.

5. Observaciones:

- Se aceptará el uso de bibliotecas de estructuras de datos, *parsing*, hilos, *hashing*, etc.

Control Intermedio

El control intermedio se realizará en el monitoreo de la semana del 03/10 y en él se deberá presentar el código de la aplicación servidor con las siguientes funcionalidades: aceptar conexiones de clientes, atender clientes de forma simultanea, ejecutar las solicitudes de los clientes y almacenar datos en una estructura de datos. Para este control intermedio se considera el caso de un único servidor por lo que no será necesario el descubrimiento de otros servidores ni la distribución de contenidos. Como cliente será suficiente usar la utilidad `telnet`. Por detalles, consulte con su tutor.

Referencias y Bibliografía Recomendada

[1] Kurose, James, y Keith W. Ross. *Redes de computadoras*. Vol. 7. Pearson Educación, 2017
[2] Comer, Douglas, *Computer Networks and Internets*, 5th Edition.
[3] Linux man-pages. https://www.kernel.org/doc/man-pages/
[4] Python Sockets Interface. https://docs.python.org/3/library/socket.html

Anexo 1

Ejemplo de ejecución del sistema:

1. Se levanta un servidor A en un host 192.168.0.100, puerto 2022. Calcula su propia firma (0x6DF87E77)
2. Se levanta un servidor B en un host 192.168.0.101, puerto 2022. Calcula su propia firma (0xA1527EE9)
3. Servidor A recibe un mensaje del protocolo DESCUBRIMIENTO desde el servidor B. Lo registra y calcula su firma (0xA1527EE9). Abre una conexión al puerto indicado para el protocolo DATOS.
4. Servidor B recibe un mensaje del protocolo DESCUBRIMIENTO desde el servidor A. Lo registra y calcula su firma (0x6DF87E77). Abre una conexión al puerto indicado para el protocolo DATOS.
5. Un cliente se conecta al servidor A y realiza la operación del protocolo DATOS “SET clave1 valor1”.
6. Servidor A calcula la firma de la clave “clave1” (0xF1011E32).
7. Busca la firma del servidor más cercana a la firma de la clave, y determina que es el servidor B (0xF1011E32 está más cerca de 0xA1527EE9 que de 0x6DF87E77)
8. Como el servidor responsable de la clave es otro, A le envía la operación del protocolo DATOS “SET clave1 valor1”.
9. Servidor B calcula la firma de la clave “clave1” (0xF1011E32).
10. Busca la firma del servidor más cercana a la firma de la clave, y determina que es él mismo (0xF1011E32 está más cerca de 0xA1527EE9 que de 0x6DF87E77).
11. Como B es el responsable de la clave, registra el valor en su almacenamiento, y responde con el mensaje del protocolo DATOS “OK”.
12. A recibe la respuesta, A le transmite a su vez la respuesta al cliente: “OK”.
13. El cliente recibe la respuesta, y como es “OK” imprime “éxito!”, y termina.