Trabajo Práctico N°2

75.59 - Técnicas de Programación Concurrentes I

Facultad de Ingeniería de la Universidad de Buenos Aires

2do. Cuatrimestre 2017

Federico Baliña, *Padrón Nro. 96.945*

federicobalina@gmail.com

Pablo Rodrigo Ciruzzi, *Padrón Nro. 95.748*

p.ciruzzi@hotmail.com

Tomás Mussi, *Padrón Nro. 94.485*

tomasmussi@gmail.com

30 de noviembre de 2017

Índice

1. Hipótesis

- El rango del tipo de dato *long* alcanza para utilizar identificadores en la asignación que realiza el servidor.

2. División en Procesos

El programa se dividió en un proceso servidor, que a su vez despacha dos procesos de servicios tanto de monedas como de estado del tiempo, n procesos cliente que se conectan al servidor para realizar consultas (Dentro de los cuales puede haber procesos de administración para la actualización de datos) y, además, por cada cliente que se conecta, el servidor despacha un proceso *worker* para atender la consulta y poder seguir atendiendo nuevos clientes.

2.1. Main

El código del programa está integrado en un mismo ejecutable, por lo que en el *main* se distingue si se quiere ejecutar el programa en modo cliente, servidor o administrador. Dependiendo de los parámetros de entrada, se distinguen las opciones:

- Cliente: *'c'*
- Servidor: *'s'*
- Administrador: *'a'*

2.2. Cliente

El cliente es el proceso que se ejecuta para la consulta del estado del tiempo de una ciudad, o bien para la consulta del tipo de cambio de una moneda en particular respecto del peso *ARS*. Para realizar una consulta se debe ejecutar el programa y, además de especificar que se quiere modo cliente, se debe especificar el tipo de consulta (Si es tiempo -1- o moneda -2-) y, como tercer parámetro, especificar en una cadena de texto la ciudad o moneda a consultar. Como se mencionó previamente, se pueden tener varios clientes simultáneos en ejecución.

2.3. Servidor

El servidor es el proceso que obtiene las peticiones de los clientes y las despacha en procesos *worker* para resolver la consulta. Las consultas se reciben en una cola de mensajes a través de un protocolo establecido entre cliente y servidor para tener una comunicación bidireccional y se realiza la consulta al servicio que corresponda para devolver una respuesta al cliente. Además, al comienzo de la ejecución, el servidor despacha los dos procesos servicio en los cuales los *workers* se apoyan para resolver las consultas.

2.4. *Worker*

Como su nombre lo indica, es el proceso en el cual el servidor deriva la conexión con el cliente para poder seguir aceptando nuevos clientes y estableciendo conexiones. El *worker* es el que hace la consulta en una nueva cola de mensajes hacia los servicios correspondientes dependiendo de la consulta recibida, y respondiendo al cliente correspondiente con la información solicitada.

2.5. Administrador

El administrador es el proceso que se encarga de actualizar la información que tienen los servicios de consulta. Es un programa por consola que permite realizar múltiples actualizaciones de valores. Actúa de forma análoga a un cliente, pero con la intención de escribir en vez de leer.

2.6. Servicios

Son dos procesos que reciben peticiones a través de una cola de mensajes (Que "comparte" con los *workers*), tanto las consultas de clientes como las actualizaciones del administrador, del estado del tiempo o de la tasa de cambio de las monedas. Cuando el servidor finaliza, señaliza a los servicios para su terminación y éstos escriben en disco toda la información que fue cargando el administrador. Esta información es luego cargada desde dicho archivo al iniciar nuevamente cada servicio.

3. Esquema de Comunicación

3.1. Servidor

El servidor funciona hasta que se le envía una señalización de interrupción (*SIGINT*) en cuyo caso se reenvía la señal a los procesos servicios para que puedan almacenar en disco la información que tienen en memoria y finalizar. La finalización queda en espera hasta que terminen todos los *workers* pendientes y los dos servicios. En el mientras tanto, ya no se reciben nuevas conexiones.

3.2. Cliente - Servidor

Esta comunicación posee una fase de establecimiento de la conexión. Para ello, el cliente envía un mensaje al servidor con un *mtype = 1* indicando que quiere comenzar una consulta, a lo que el servidor le responde a través de la misma cola con un mensaje que posee un *mtype = 2* y donde se le indica en el campo *id* cuál es el id de cliente. A partir de ese momento, el servidor creó un *worker* específico para ese cliente, y al cual se puede le puede hacer una única consulta utlizando la misma cola de mensajes pero en este caso con un *mtype = <id-cliente>*. Por último el *worker* responderá la consulta utilizando un mensaje con *mtype = <id-cliente> + 1*.

Para una descripción más gráfica de dicho protocolo ver la sección 5.3.

3.3. Servidor - Servicios

Análogamente a lo descrito entre cliente y servidor, cada *worker* que recibe una petición de un cliente, accede al servicio correspondiente a través de otra cola de mensajes. Éste escribirá la consulta utilizando el *mtype* correspondiente al servicio al cual se quiere consultar (1 para el de tiempo, 2 para el de monedas), junto con el campo *id* = <*id-cliente*>. De esta manera, el servicio responderá la consulta con un mensaje con *mtype* = <*id-cliente*>, de forma de que el *worker* sepa a qué tipo de mensaje debe quedarse esperando.

Para una descripción más gráfica de dicho protocolo ver la sección 5.3.

4. Mecanismos de concurrencia

4.1. Señales

El único evento asincrónico con necesidad de señalizar es la finalización de ejecución del servidor con *SIGINT (2)*. Cuando se envía esta señal, el servidor a su vez se la envía a dos de sus procesos hijos que son los servicios. Los servicios guardan en disco la información que tienen en memoria y finalizan correctamente, al igual que el servidor. Asímismo, esta señal no es redirigida hacia los *workers*, haciendo que el servidor no termine hasta que no se resuelvan todas las consultas. De todas formas, una vez que el servidor recibió esta señal, ya no se reciben más conexiones.

4.2. Cola de mensajes (*Queues*)

Se utilizaron dos colas de mensajes distintas:

- **Cliente - Servidor:** se utilizan mensajes de tipo:
  - 1: Nueva conexión
  - 2: Respuesta a nueva conexión
  - De 3 en adelante: ID e ID + 1 asignado por el servidor con la conexión establecida.

  Con esta cola de mensajes se logran comunicar el cliente y el servidor, teniendo un protocolo de "bienvenida" al estilo *3-way handshake* de TCP/IP. No existe un protocolo de finalización, ya que cada cliente puede ejecutar una única consulta.

- **Servidor - Servicios:** se utilizan mensajes de tipo:
  - 1: Tiempo
  - 2: Moneda
  - De 3 en adelante: ID. Con el ID asignado por el servidor al cliente, se utiliza este identificador para que los servicios envíen por la misma cola de mensajes al *worker* del cliente correspondiente la respuesta a su consulta.

5. Diagramas

5.1. Diagrama de procesos e IPC

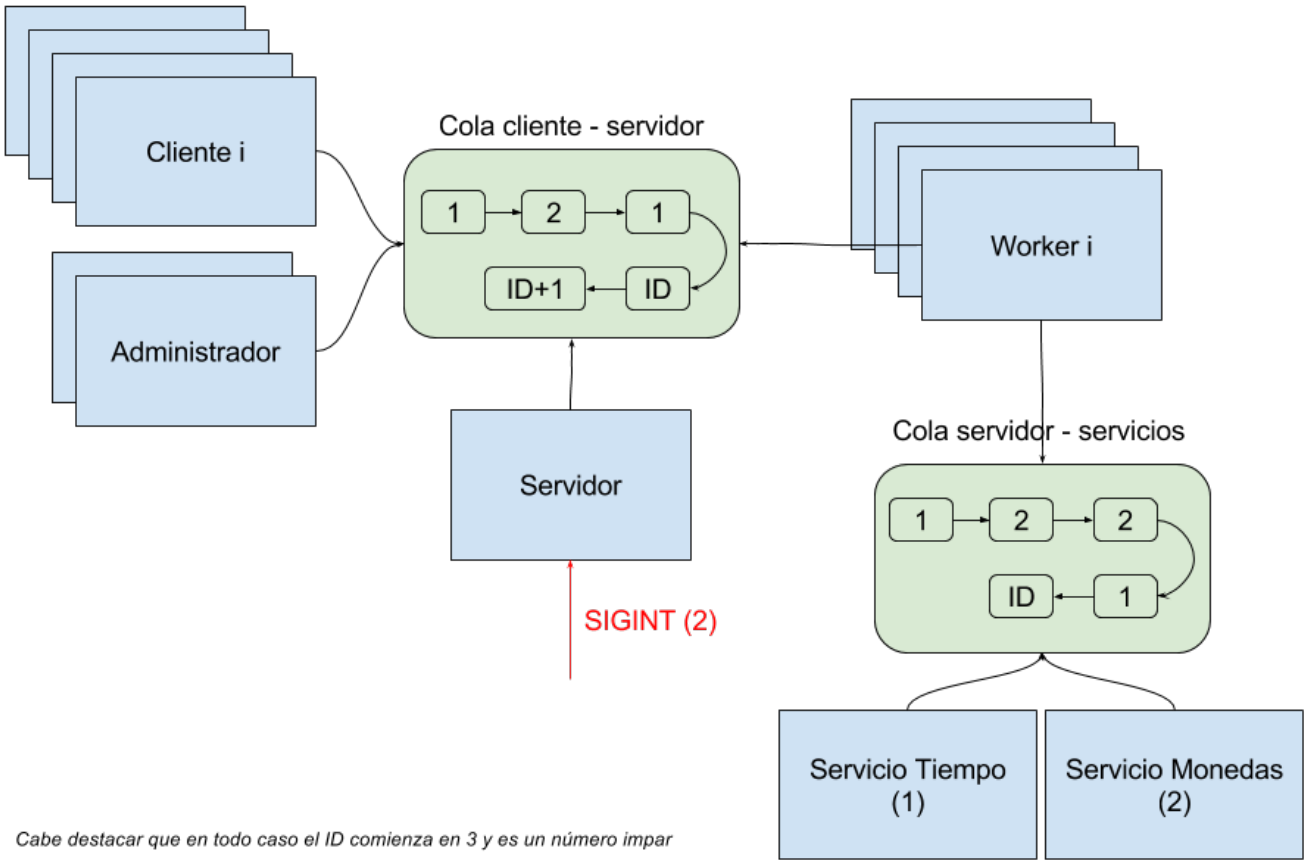

Figura 1: Diagrama de procesos e IPC

5.2. Diagramas de clases

A continuación se muestra el diagrama de clases del programa.

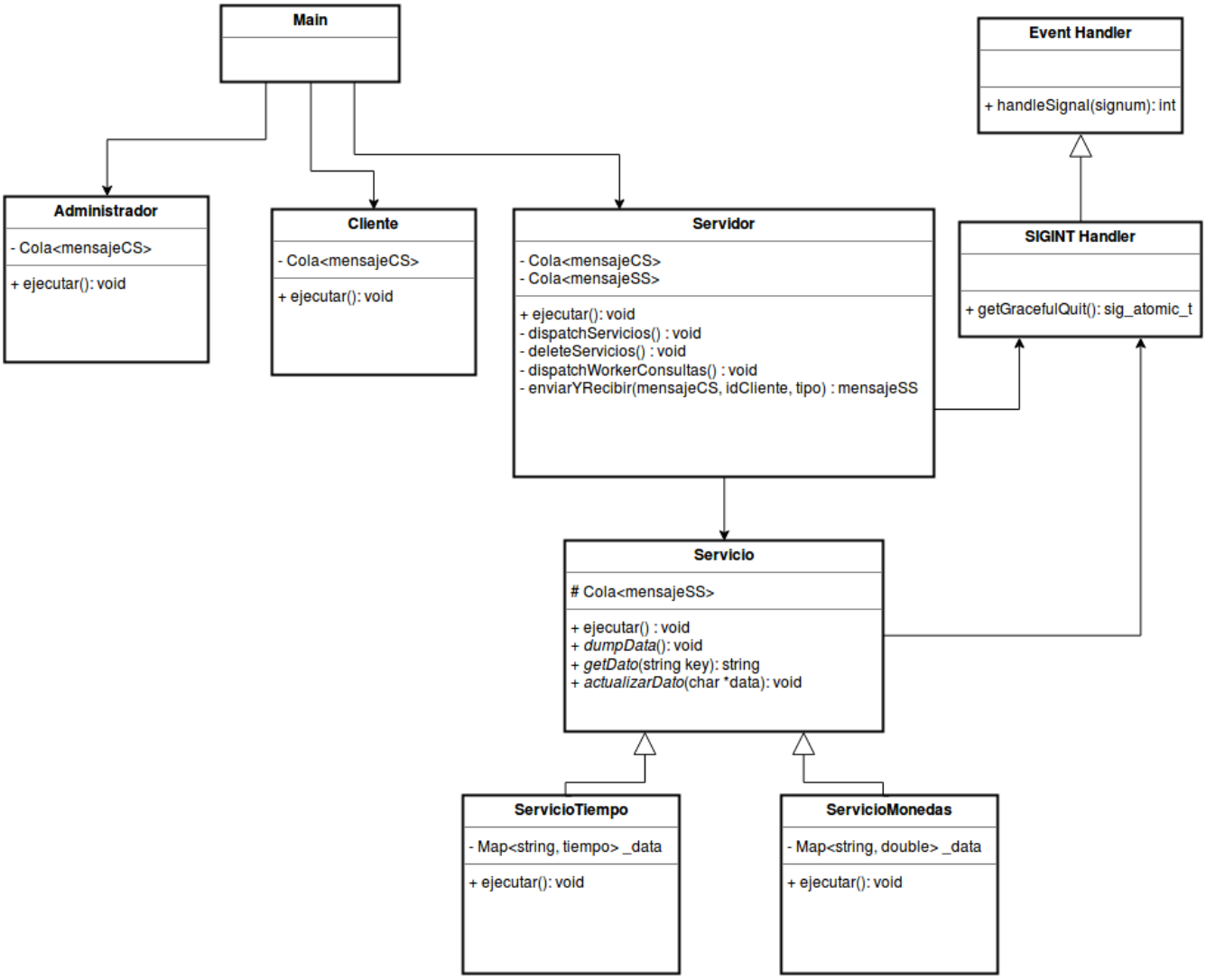

Figura 2: Diagrama de clases del TP

5.3. Diagrama de secuencia de petición del cliente

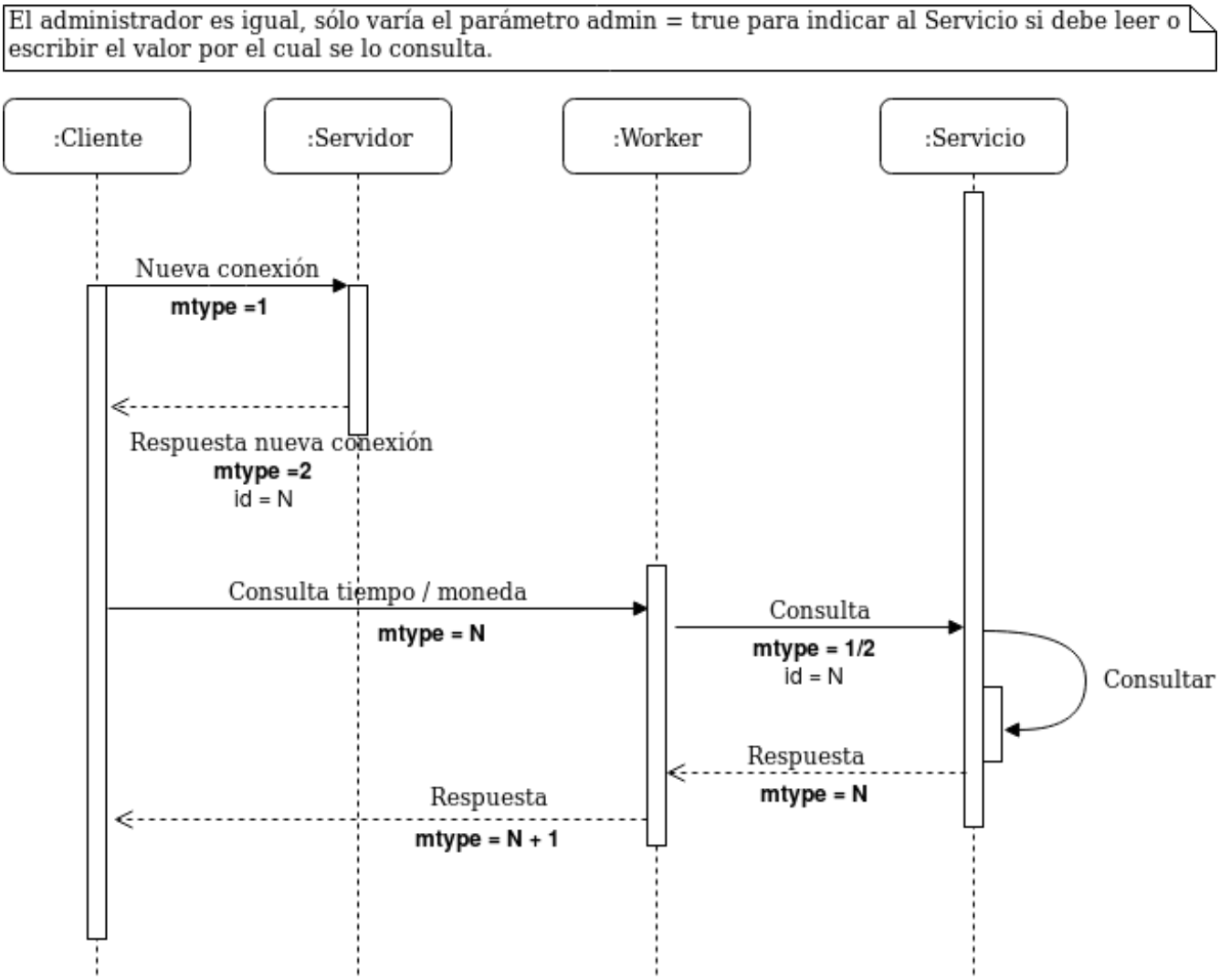

Figura 3: Diagrama de secuencia de consulta de cliente

Es importante destacar que el servidor devuelve en *ID* sólo valores impares, de forma de dejar libre el *ID + 1* para la respuesta. De esta manera, los *ID* impares son consultas y los pares son respuestas.

6. Instrucciones

Si se ejecuta el programa sin parámetros, se mostrará un cartel de ayuda que indica cómo ejecutar el cliente, el servidor y el administrador.