Sistemas Distribuidos

zeroC-system

Samuel González Linde

Repositorio:
https://github.com/Samuglz6/zeroC-system

Escuela Superior de Informatica
Universidad de Castilla-La Mancha

June 2020

Contents

1 Introduction

[1]

El objetivo principal del proyecto es diseñar un sistema cliente-servidor que permita la descarga de ficheros. La implementación de este proyecto permitirá al alumno trabajar, mediante ZeroC Ice, los siguientes aspectos:

- Transparencia de localización
- Manejo de canales de eventos
- Despliegue de servidores
- Arquitectura del proyecto

1.1 Descripcion del Sistema

El sistema estará compuesto por un cliente llamado FileDownloader con una factoría de receivers,un servidor con una factoría de transfers y otro servidor con una factoría de senders. El cliente, querecibirá como argumentos el nombre de los ficheros, solicitará una transferencia de N ficheros alservidor donde están ubicados los transfers. Este creará un transfer para controlar la transferencia,que a su vez creará una pareja receiver-sender para cada uno de los ficheros. Cuando todas lasparejas estén listas el cliente iniciará la transferencia, es decir, el envío entre las parejas. Losreceivers notificarán al transfer cuando hayan terminado, y este destruirá al receiver que envía lanotificación y a su sender compañero. Si todas las parejas han terminado, el tranfer informará alcliente (estará a la espera), que destruirá el transfer y terminará su ejecución.

También será importante el control de fallos, al menos de los que pueden tener lugar por accionesdel cliente, para garantizar el correcto funcionamiento del lado del servidor. Por ejemplo, en el casode que se hagan solicitudes de ficheros que no existen la transferencia se daría por fallida, inclusocuando solamente uno de los ficheros solicitados no existiese, destruyendo todos los componentesrelacionados con la transferencia e informando al cliente de lo sucedido.

2 Arquitectura del programa

El sistema va a estar formado por cinco tipos de componentes:

- **senders:** Se van a encargar del envío de los ficheros solicitados.
- **transfers:** Se encargaran de la creacion y destrucción de los elementos necesarios para la transferencia del archivo.
- **receivers:** Encargados de solicitar la recepción de archivos.

[1] *El contenido de la Introducción y Descripción del Sistema viene dado por el enunciado de la práctica proporcionado por la UCLM.*

- **clientes:** Serán quienes soliciten los ficheros a descargar.
- **canales de eventos:** Se emplearán para la comunicación del estados entre los componentes del sistema.

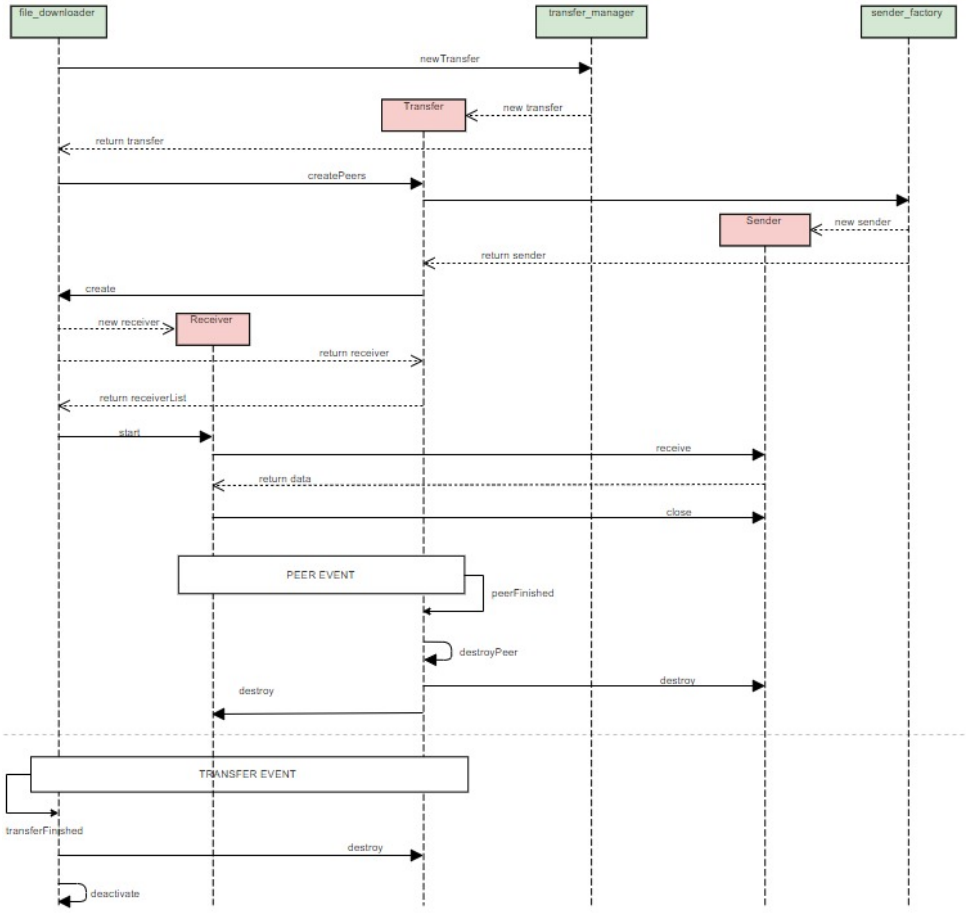

Figure 1: Diagrama de secuencia

3 Estructura del proyecto

En lugar de contener todos y cada uno de los archivos en la carpeta raiz del proyecto, se ha creado una estructura de directorios para mantener organizados los archivos del proyecto:

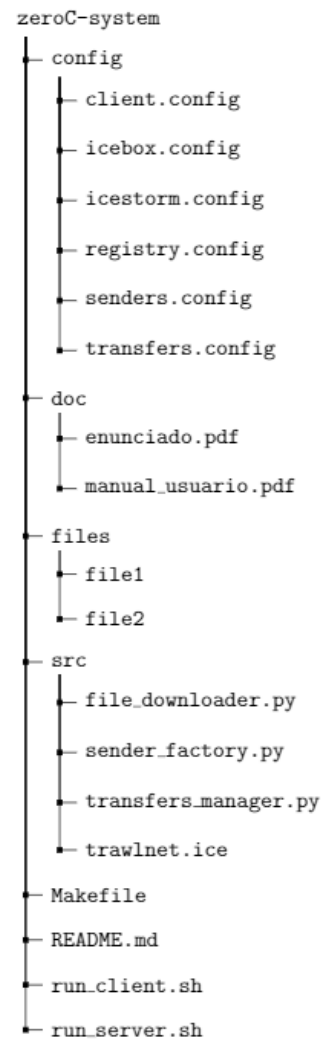

Figure 2: Estructura del proyecto

También hay que añadir que durante la ejecución del programa una serie de nuevos directorios serán creados en el proyecto. Mediante la ejecución del comando make clean en el Makefile, podremos eliminarlos.
Los directorios que serán creados son los siguientes:

- **downloads:** Directorio que almacenará las descargas solicitadas por el Cliente. Será creado durante la ejecución del Cliente.
- **db:** Aqui serán registrados los adaptadores y los objetos. Se creará durante la ejecución del registry.
- **IceStorm:** Se registrarán los canales de eventos para el uso de las comunicaciones. Será creado durante la ejecución del IceStorm.

4 Archivos de ejecución

4.1 Scripts

4.1.1 run_server.sh

Se encarga de crear los directorios necesarios para el uso de los servicios del Registry y el IceStorm, de la ejecución de dichos servicios y finalmente de la ejecución de la parte del servidor, compuesta por la ejecución del sender_factory.py seguida de la ejecución de transfers_manager.py

```
$ ./run_server.sh
```

4.1.2 run_client.sh

Se encarga de la creación del directorio de descargas donde van a ser almacenados los ficheros solicitados y de la ejecución del cliente utilizando como parámetros para el cliente los mismos parámetros que le pasemos a run_client.sh

```
$ ./run_client.sh <archivo1> <archivo2>...
```

4.2 Archivo MAKEFILE

Dentro del archivo Makefile disponemos de diferentes comandos que nos van a ser de gran utilidad para poder ejecutar algunas de las partes del proyecto y ver su ejecución por separado.

4.2.1 Ejecución del Servidor

Comando que ejecuta todo lo necesario para que el lado del servidor se despliegue (registry, icestorm, sender factory y transfer manager):

```
$ make run-registry
```

4.2.2 Activación del registry

Comando de activación del registry que nos proporciona transparencia de localización:

```
$ make run-registry
```

4.2.3 Activación del IceStorm

Comando de activación para el uso del servicio de canales de eventos de IceStorm:

```
$ make run-icestorm
```

4.2.4 Ejecucion del sender_factory

Comando para la ejecución de la factoría de senders:

```
$ make run-sender-factory
```

4.2.5 Ejecucion del transfer_manager

Comando para la ejecucion del Gestor de transferencias:

```
$ make run-transfer-manager
```

4.2.6 Ejecucion del Cliente

Comando para la ejecución del cliente (por defecto los argumentos utilizados como ficheros para la descarga son file1 y file2):

```
$ make run-client
```

4.3 Creación del directorio de descargas

Comando para la creación del directorio donde el usuario recibirá el archivo solicitado:

```
$ make create-client-workspace
```

4.3.1 Limpiar el directorio del proyecto

Comando para eliminar los directorios que se utilizan durante la ejecución del programa:

```
$ make clean
```

5 Ejecución del programa

Para ejecutar el proyecto podemos hacerlo de dos formas distintas: utilizando dos cripts o utilizando un Makefile En ambos casos nos situaremos en la raiz de la carpeta del proyecto para poder utilizarlos.

```
$ cd /zeroC-system
```

También hay que tener en cuenta, que para poder desplegar de nuevo el sistema, es necesario que al finalizar la ejecución del lado del servidor (Interrumpiendo su ejecución con Ctrl+C) es necesaria la ejecucion del comando del MAKEFILE 'clean' para poder despejar los servicios de registry y IceStorm.

5.1 Ejecución usando MAKEFILE

Tenemos también dos formas de hacerlo: ejecutando todo por separado o ejecutando el lado del servidor y el del cliente.

5.1.1 Ejecución por separado

Primero tendremos que ejecutar el registry y el IceStorm para poder hacere uso de estos servicios, por lo tanto:

```
$ make run-registry
$ make run-icestorm
```

Comenzamos ejecutamos el lado del servidor, empezando por la factoría de los senders ya que es necesario el registro de su adaptador para que el gestor pueda utilizarlo:

```
$ make run-sender-factory
```

Y ponemos en marcha el gestor de las transferencias:

```
$ make run-transfer-manager
```

Una vez ambos programas del lado del servidor estén corriendo, hacemos ejecución del servidor:

```
$ make run-client
```

Finalmente una vez ha finalizado el programa podemos interrumpir la ejecución del lado del servidor pero para poder volver a ejecutarse es necesaria la limpieza de los directorios de los servicios:

```
$ make clean
```

5.1.2 Ejecución de ambos lados

Podemos ejecutar directamente el lado del servidor y el lado del cliente. Tenemos que tener en cuenta que para que el cliente funcione, el lado del servidor tiene que estar desplegado primero. Por lo tanto, primero realizamos la activacion del servidor:

```
$ make run-server
```

Esto se encargará de alzarnos el registry, el icestorm, la factoria de senders y el gestor de transferencias.
Una vez esta todo en marcha procedemos a la ejecucion del cliente:

```
$ make run-client
```

Finalmente una vez ha finalizado el programa podemos interrumpir la ejecución del lado del servidor pero para poder volver a ejecutarse es necesaria la limpieza de los directorios de los servicios:

```
$ make clean
```

5.2 Ejecución usando Scripts

Primero vamos a realizar la ejecución del lado del servidor y todas sus componentes necesarias. Para ello hacemos uso del script run_server.sh.

```
$ ./run_server.sh
```

Después ejecutamos el lado del cliente haciendo uso del script run_client.sh y pasandole como argumentos el nombre del archivo o archivos que quieren ser descargados.

```
$ ./run_client.sh <archivo1> <archivo2>...
```

Finalmente una vez ha finalizado el programa podemos interrumpir la ejecución del lado del servidor pero para poder volver a ejecutarse es necesaria la limpieza de los directorios de los servicios:

```
$ make clean
```

6 Referencias

- Enunciado del proyecto
- Manual de ZeroCIce de Moodle UCLM
- Ice 3.7 Slice API Reference