Universidad Simón Bolívar

GGGOM
Geodistributed Getter Of Movies

Gustavo El Khoury - 10-10226
Gabriela Limonta - 10-10385
Moisés Ackerman - 11-10005
Gustavo Gutiérrez - 11-10428
Oscar Guillen - 11-11264

Introducción

GGGOM (*Geodistributed Getter Of Movies*) es una aplicación de tipo cliente/servidor que involucra tres actores:

- **SC** - Un servidor central de registro de usuarios y control de descargas.
- **SD** - Varios servidores de descarga que atienden a los clientes.
- **CL** - Clientes que solicitan películas.

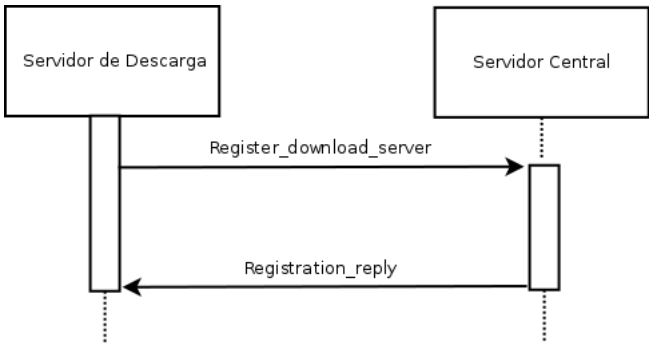
La intención de este trabajo es explicar las decisiones de diseño que se tomaron a la hora de implementar este trabajo (definición del protocolo, herramientas a utilizar) así como a describir brevemente la estructura del código implementado.

Definición del protocolo

Se especificarán los protocolos de comunicación entre cada uno de los actores.

Mensajes entre SC y SD

1. El servidor de descarga se registra en el servidor central.

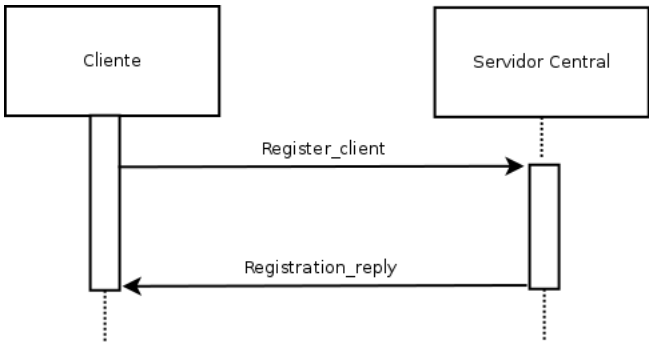


   Lenguaje de mensajes a utilizar:

   - Register_download_server: *Header del mensaje. Contiene los siguientes elementos:*
     - Host: *Direccion del servidor de descarga.*
     - Port: *Puerto de escucha del servidor de descarga.*
     - Movie: *Una o varias películas que contiene el servidor.*
       - id_movie: *Id de la película.*
       - title: *Título de la película.*
       - size: *Tamaño de la película.*
   - Registration_reply: *Header del mensaje. Contiene los siguientes elementos:*
     - Reply: *Respuesta del servidor. Puede ser* ***Ok*** *o* ***Failed****.*

Mensajes entre SC y CL

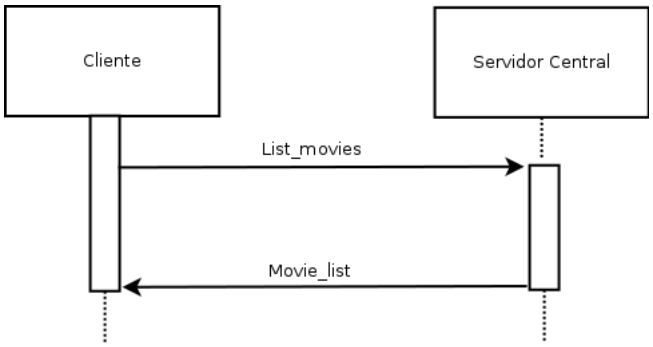
1. El cliente se registra en el servidor central.



   Lenguaje de mensajes a utilizar:

   - Register_client: *Header del mensaje. Contiene los siguientes elementos:*
     - Host: *Direccion del cliente.*
     - Port: *Puerto de conexión del cliente.*
     - Username: *Usuario con el cual se va a registrar. No puede estar repetido.*
   - Registration_reply: *Header del mensaje. Contiene los siguientes elementos:*
     - Reply: *Respuesta del servidor. Puede ser* ***Ok*** *o* ***Failed****.*

2. El cliente le pide la lista de películas disponibles al servidor central. Debe estar previamente registrado.



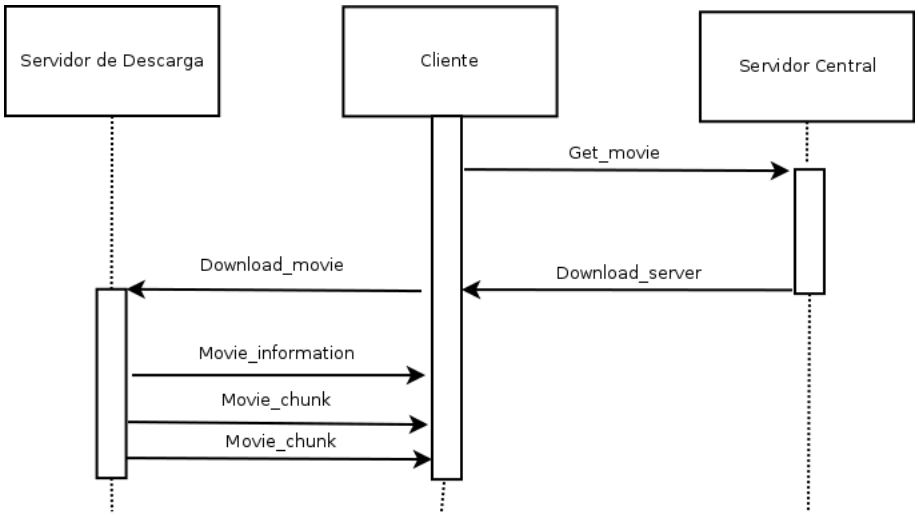
   Lenguaje de mensajes a utilizar:

   - List_movies: *Header del mensaje. No contiene elementos.*

- Movie_list: *Header del mensaje. Contiene una lista de películas con el siguiente formato:*
  - id_movie: *Id de la película.*
  - title: *Título de la película.*
  - size: *Tamaño de la película.*

Mensajes que involucran a los tres actores.

1. El cliente le pide una película al servidor central. Luego descarga la película del servidor de descarga adecuado.



Lenguaje de mensajes a utilizar:

- Get_movie: *Header del mensaje. Contiene los siguientes elementos:*
  - Username: *Usuario con el se va a descargar la película. Debe estar registrado.*
  - Id_movie: *Película que se desea descargar.*
- Download_server: *Header del mensaje. Contiene los siguientes elementos:*
  - Server: *Información del servidor de descarga. Tiene el siguiente formato:*
    - Host: *Dirección del servidor de descarga.*
    - Port: *Puerto del servidor de descarga.*
- Download_movie: *Header del mensaje. Contiene los siguientes elementos:*

    - Id_movie: *Id de la película a descargar.*
- Movie_information: *Header del mensaje. Contiene los siguientes elementos:*
    - File_name: *Nombre del archivo pedido.*
    - File_size: *Tamaño del archivo pedido.*

Implementación

La implementación del proyecto se decidió realizar en *Python*. Esto se debe a varias razones entre las cuales destacan la familiaridad de los desarrolladores con el lenguaje, las bondades de los lenguajes orientados a objetos y la amplia documentación con la que cuenta *Python*.

Librerías utilizadas

Twisted

Twisted es una librería para el desarrollo de aplicaciones de red. Contiene una gran cantidad de protocolos, servidores y clientes listo para ser usados, así como también incluye una interfaz para la creación de protocolos a la medida del usuario.

En el contexto de **GGGOM** se utilizó para la definición de los protocolos a implementar y el manejo de las conexiones.

Cmd

Cmd es un modulo incluido en python que provee un marco de desarollo para la escritura de interpretadores de comandos.

Se utilizó en el proyecto para la implementación de los interpretadores de comandos de cada uno de los actores.

Tabulate

Tabulate es una librería de python que facilita la impresión de estructuras de datos complejas por pantalla.

Se utilizó para imprimir los resultados de los comandos que involucran estructuras complejas como lo son la lista de películas o la lista de clientes.

Estructura del código

Twisted

Antes de hablar de la estructura del código implementado es conveniente aclarar un poco la interfaz especificada por Twisted. Para la definición de un protocolo con Twisted es necesario implementar un par de clases. La primera es una clase que herede de *Protocol* y es donde se definirán las acciones a realizar para cada conexión mediante ese protocolo. La segunda clase debe heredar de *Factory* y será la clase que cree cada una de las instancias del Protocolo y mantiene información persistente entre instancias.

Archivos por actor

Para cada uno de los actores se definieron 4 archivos llamados *Actor*, *Actor*.py, *Actor*_service.py y *Actor*_factory.py. Por ejemplo, para el cliente existen los archivos **client**, **client.py**, **client_service.py** y **client_factory.py**. La función de cada archivo es la siguiente:

- ***Actor*:** es un script sencillo que lee los argumentos de pantalla e instancia las clases iniciales.
- ***Actor*.py:** Contiene la definición del interpretador así como la clase que representa al actor.
- ***Actor*_service.py:** define una interfaz que le permite al interpretador invocar a los métodos del protocolo de una manera más sencilla.
- ***Actor*_factory.py:** contiene las definiciones de los protocolos que involucran al actor y sus clases factory correspondientes.

Archivos comunes

- **client_item.py:** Contiene la clase *ClientItem* que define a un cliente del sistema.
- **server_item.py:** Contiene la clase *ServerItem* que define a un servidor.
- **common.py:** Contiene la definición de clases auxiliares.
- **movie.py:** Contiene la definición de la clase *Movie* que representa a una pelicula.
- **request.py:** Contiene la definición de la clase *Request*.