



ZeroC

YOUTUBE

DOWNLOADER

Entrega L3

DESCRIPCIÓN BREVE

Aplicación implementada con ZeroC Ice de un sistema cliente-servidor distribuido que permite la descarga de ficheros a partir de URIs.

LUCÍA ALFONSO GARCÍA

EDUARDO GARCÍA APARICIO

SSDD – ESI - UCLM

Tabla de contenido

1.	Descripción de los archivos	2
2.	Despliegue de la aplicación	3
3.	IceGridGui	4

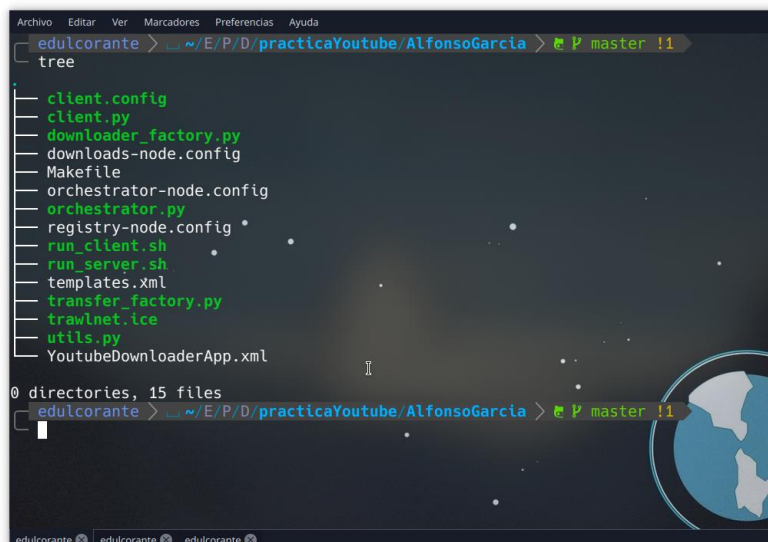
Tabla de ilustraciones

Ilustración 1	Árbol de la aplicación	3
Ilustración 2	Make run	3
Ilustración 3	run_server.sh	4
Ilustración 4	Crear nueva conexión	5
Ilustración 5	Abrir aplicación	5
Ilustración 6	Distribuir aplicación	6
Ilustración 7	run_client.sh	6
Ilustración 8	run-client-download.....	7
Ilustración 9	run-client-transfer	7
Ilustración 10	run-client-list.....	8
Ilustración 11	Retrieve stdout.....	9

1. Descripción de los archivos

En el directorio principal de la práctica podemos encontrar los siguientes archivos:

- **client.config:** contiene la dirección del *locator* para poder localizar a los *orchestrator*.
- **client.py:** contiene todos los métodos necesarios para descargar y transferir la lista de archivo.
- **downloader_factory.py:** contiene el *servant downloader* y el código para descargar los archivos, así como la factoría de *downlaoders*.
- **downloads-node.config:** contiene la dirección del *locator*, del *TopicManager* y la configuración del nodo de descargas.
- **makefile:** define el conjunto de tareas a ejecutar, tanto para iniciar el cliente como el *orchestrator*, los *downloads* o el *registry*.
- **orchestrator-node.config:** contiene la dirección del *locator* y del *TopicManager*, así como la configuración de los *orchestrators*.
- **orchestrator.py:** contiene el sirviente del *orchestrator* y establece las comunicaciones con el transfer y *downloaderFactory*. Crea los canales de *UpdateEvent* y *OrchestratorSync* y establece los *publisher* y *subscriber* en cada canal. Además de esto, recibe las peticiones de los clientes y se las transfiere tanto al *downloaderFactory* y *TransferFactory*, y en caso de que el cliente le pida la lista de canciones descargadas, será el propio *orchestrator* el que se encargue de leer dicha lista y enviársela al cliente.
- **registry-node.config:** contiene la dirección del *locator*, *topicManager* y el *Registry*, así como la configuración de la aplicación y el *MasterRegistry*,
- **run_client.sh:** script que ejecuta todas las opciones que tiene disponible el cliente (descargar, transferir y obtener la lista de archivos).
- **run_server.sh:** script que ejecuta al servidor inicializando todos los nodos.
- **Templates.xml:** se tuvo que añadir este archivo proporcionado ya que utilizamos una versión del IceGrid superior la cual no tiene los templates que necesitábamos.
- **Utils.py:** esta clase contiene métodos auxiliares relacionados con el manejo de archivos.
- **YoutubeDownloaderApp.xml:** aplicación que controla todos los nodos y que se desarrollará con mayor profundidad a lo largo de este manual, explicando su funcionamiento y características.



```
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P master !1
tree
.
├── client.config
├── client.py
├── downloader_factory.py
├── downloads-node.config
├── Makefile
├── orchestrator-node.config
├── orchestrator.py
├── registry-node.config
├── run_client.sh
├── run_server.sh
├── templates.xml
├── transfer_factory.py
├── trawlnt.ice
├── utils.py
└── YoutubeDownloaderApp.xml

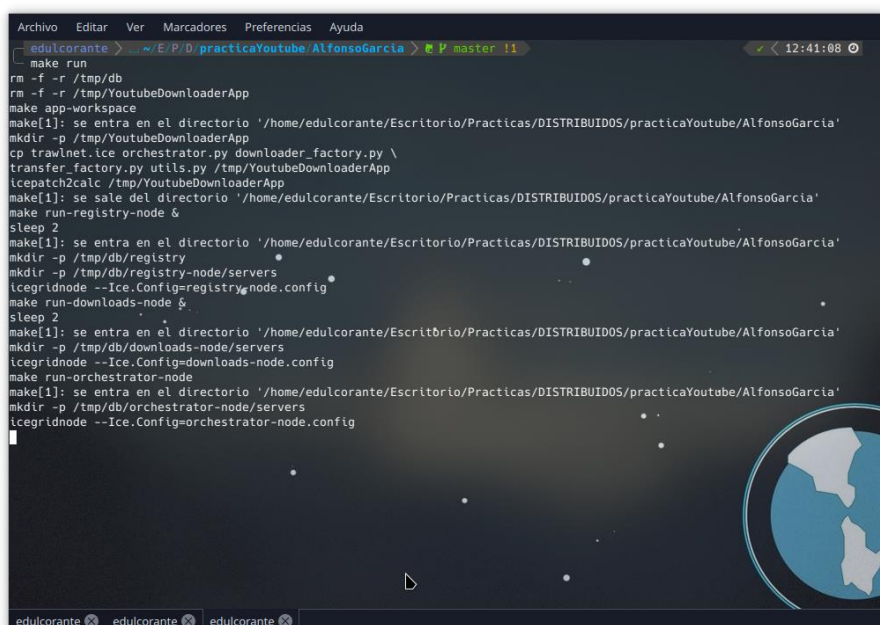
0 directories, 15 files
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P master !1
```

Ilustración 1 Árbol de la aplicación

2. Despliegue de la aplicación

Existen dos formas para comenzar a desplegar la aplicación y sus nodos:

1. Hacer un *make run*: borra todos los datos de persistencia (canciones descargadas, listas de canciones, *logs* de los nodos) y ejecuta una instancia nueva de los nodos de la aplicación.



```
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P master !1
make run
rm -f -r /tmp/db
rm -f -r /tmp/YoutubeDownloaderApp
make app-workspace
make[1]: se entra en el directorio '/home/edulcorante/Escritorio/Practicas/DISTRIBUIDOS/practicaYoutube/AlfonsoGarcia'
mkdir -p /tmp/YoutubeDownloaderApp
cp trawlnt.ice orchestrator.py downloader_factory.py \
transfer_factory.py utils.py /tmp/YoutubeDownloaderApp
icepatch2calc /tmp/YoutubeDownloaderApp
make[1]: se sale del directorio '/home/edulcorante/Escritorio/Practicas/DISTRIBUIDOS/practicaYoutube/AlfonsoGarcia'
make run-registry-node &
sleep 2
make[1]: se entra en el directorio '/home/edulcorante/Escritorio/Practicas/DISTRIBUIDOS/practicaYoutube/AlfonsoGarcia'
mkdir -p /tmp/db/registry
mkdir -p /tmp/db/registry-node/servers
icegridnode --Ice.Config=registry-node.config
make run-downloads-node &
sleep 2
make[1]: se entra en el directorio '/home/edulcorante/Escritorio/Practicas/DISTRIBUIDOS/practicaYoutube/AlfonsoGarcia'
mkdir -p /tmp/db/downloads-node/servers
icegridnode --Ice.Config=downloads-node.config
make run-orchestrator-node
make[1]: se entra en el directorio '/home/edulcorante/Escritorio/Practicas/DISTRIBUIDOS/practicaYoutube/AlfonsoGarcia'
mkdir -p /tmp/db/orchestrator-node/servers
icegridnode --Ice.Config=orchestrator-node.config
```

Ilustración 2 Make run

2. Utilizando el script `./run_server.sh`: ejecuta una instancia nueva de los nodos de la aplicación.





```
Archivo Editar Ver Marcadores Preferencias Ayuda
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P master 11
./run_server.sh
Creating directories in /tmp...
Exec icepatch2calc...
Exec registry-node
Exec downloads-node
Exec orchestrator-node
```

Ilustración 3 `run_server.sh`

Una vez que tenemos los *nodos* y el *registry* en ejecución, tenemos que ejecutar *IceGridGui*.

3. IceGridGui

Una vez iniciado el *IceGridGui* debemos conectarnos al registro  creando una conexión nueva (Ilustración 4 Crear nueva conexión). Cuando te piden un usuario y contraseña puedes poner cualquier valor, pues no está activada esa opción en la aplicación.

Después tenemos que cargar la aplicación (File → Open → Application From File) con el nombre `YoutubeDownloaderApp.xml`, tal y como se indica en la Ilustración 5 Abrir aplicación, que habrá que guardarla en registro  y distribuirla (Live Deployment → Tools → Application → Patch Distribution).

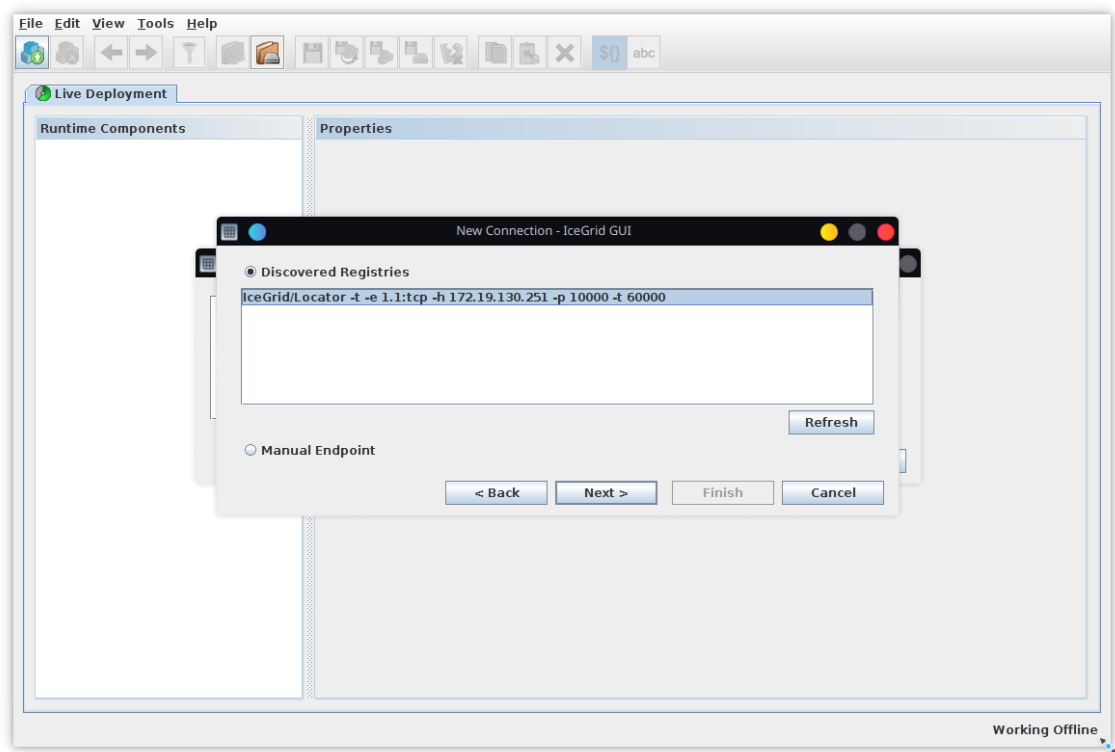


Ilustración 4 Crear nueva conexión

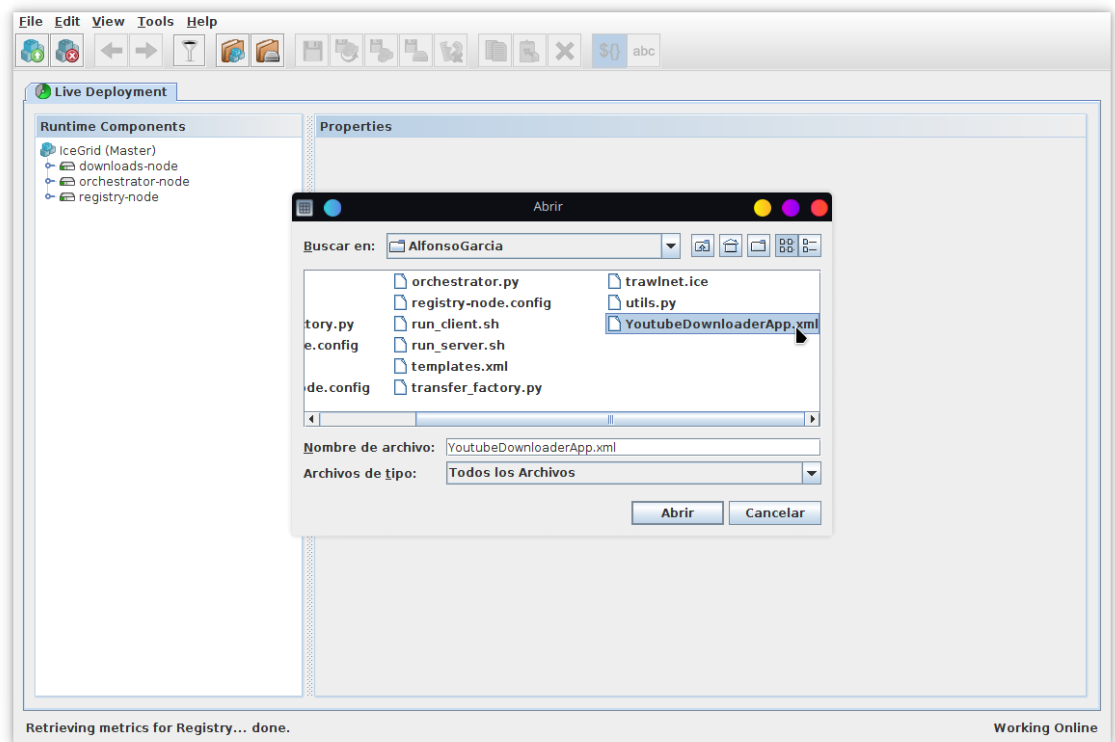


Ilustración 5 Abrir aplicación

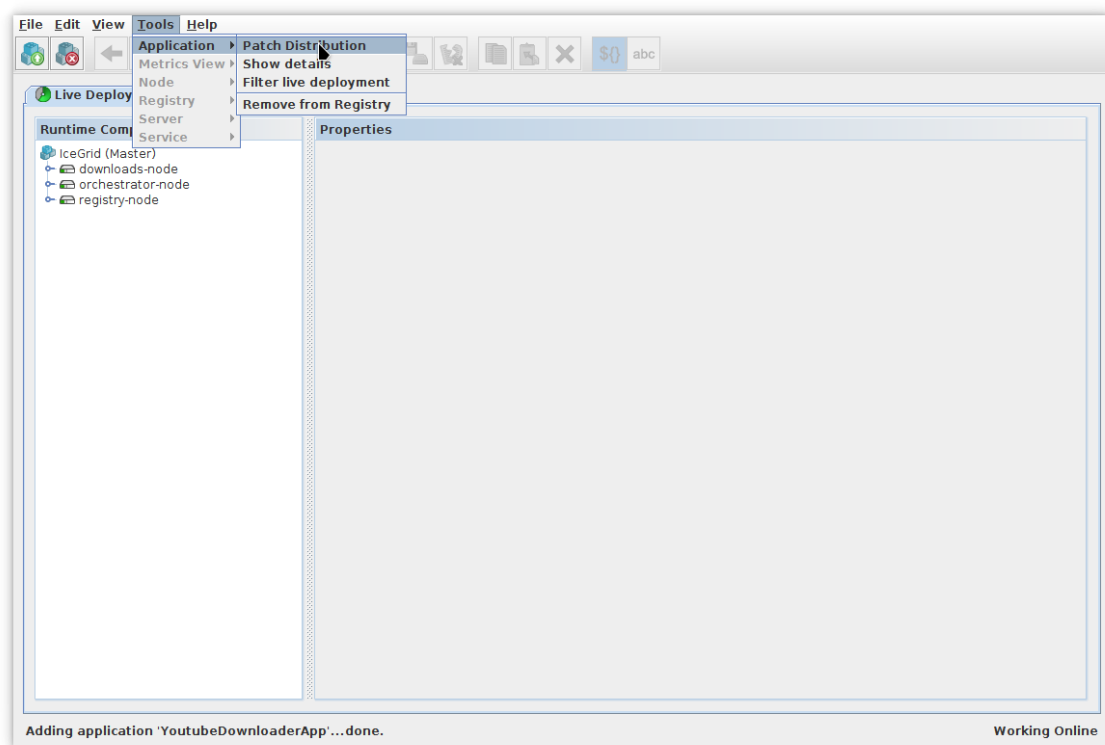


Ilustración 6 Distribuir aplicación

Una vez hecho esto, podremos ejecutar el cliente por medio del `./run_client.sh`. Podemos comprobar el resultado con la Ilustración 7 `run_client.sh`

```

Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
[edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P L3 !1]
./run_client.sh
Downloading audio...
Enviando petición a
orchestrator -t -e 1.1
Descargando: https://www.youtube.com/watch?v=9Meg1-02k5Q
[Título: Poseidón
Hash: 9Meg1-02k5Q]

List request...
Enviando petición a
orchestrator -t -e 1.1
Petición de lista de archivos mandada...
[
{
  name = Poseidón
  hash = 9Meg1-02k5Q
}]

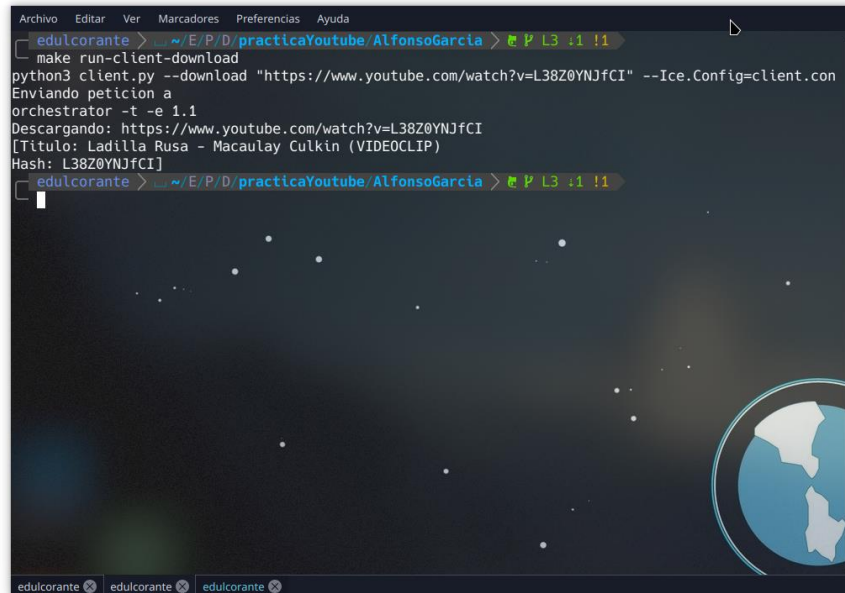
Init transfer...
Enviando petición a
orchestrator -t -e 1.1
Obteniendo: Poseidón
Transfer finished!
[edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > P L3 !1]

```

Ilustración 7 run_client.sh

Sin embargo, si optamos por la opción de ejecutarlo con el *Makefile* tendremos 3 caminos:

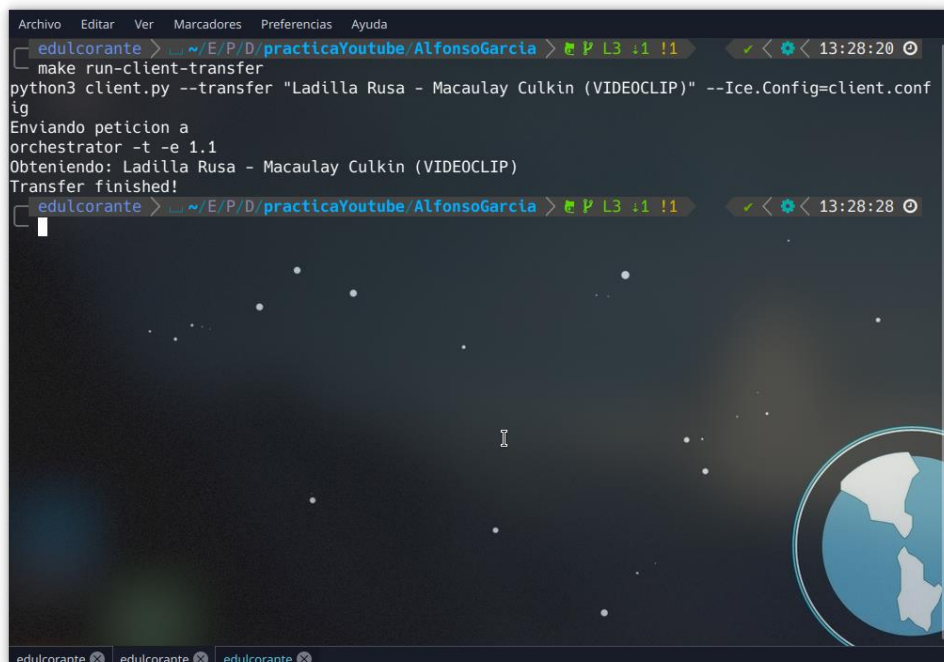
1. Ejecutar *make run-client-download*: esto nos descargará un video de Youtube o Vimeo en formato mp3 (para cambiar la canción a descargar tendrías que meterte en el código del script).



```
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > make run-client-download
python3 client.py --download "https://www.youtube.com/watch?v=L38Z0YNJfCI" --Ice.Config=client.conf
Enviando petición a
orchestrator -t -e 1.1
Descargando: https://www.youtube.com/watch?v=L38Z0YNJfCI
[Título: Ladilla Rusa - Macaulay Culkin (VIDEOCLIP)]
Hash: L38Z0YNJfCI
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia >
```

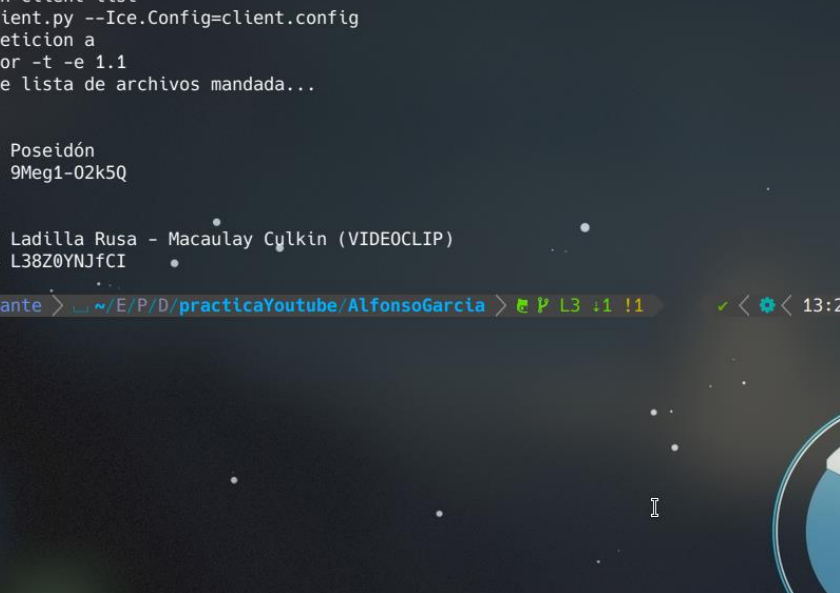
Ilustración 8 run-client-download

2. Ejecutar *make run-client-transfer*: este método inicia la transferencia de cualquier archivo descargado por la aplicación. En caso de que no se encuentre ese archivo, la aplicación finaliza.



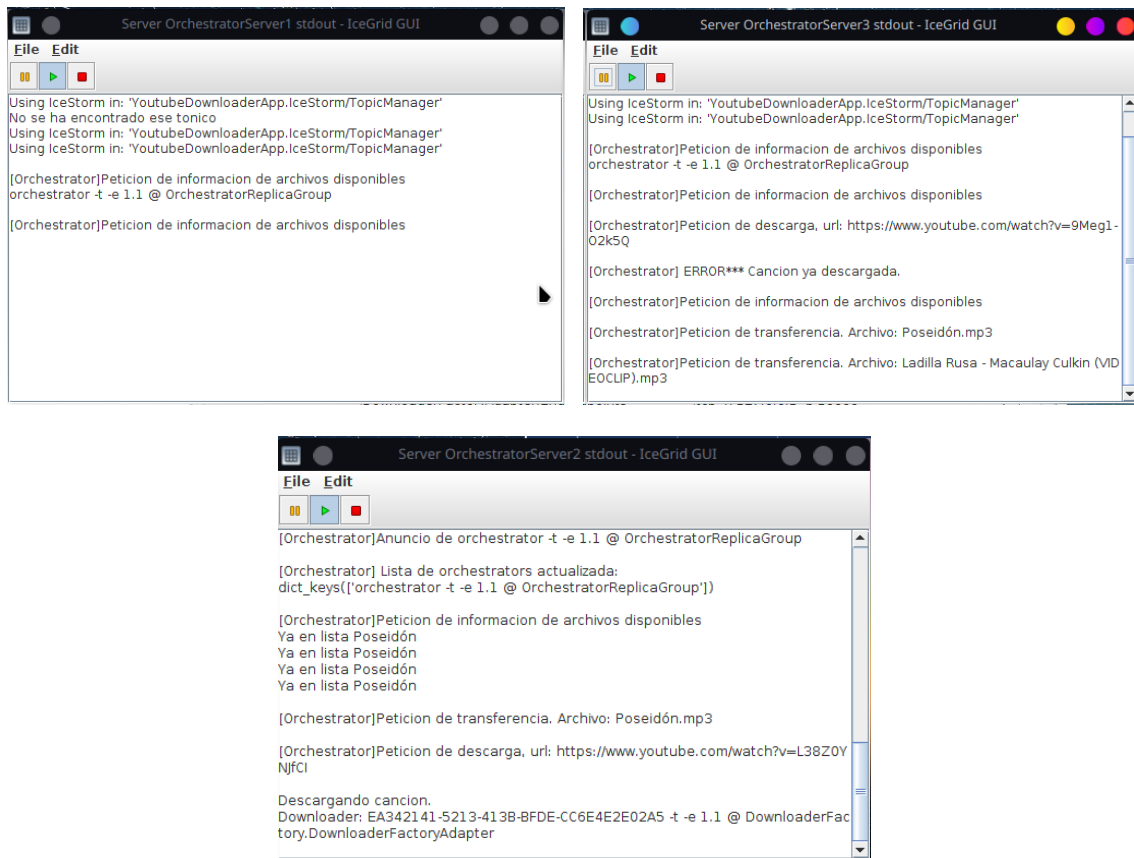
```
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia > make run-client-transfer
python3 client.py --transfer "Ladilla Rusa - Macaulay Culkin (VIDEOCLIP)" --Ice.Config=client.conf
ig
Enviando petición a
orchestrator -t -e 1.1
Obteniendo: Ladilla Rusa - Macaulay Culkin (VIDEOCLIP)
Transfer finished!
edulcorante > ~/E/P/D/practicaYoutube/AlfonsoGarcia >
```

Ilustración 9 run-client-transfer

- 
- ```
Archivo Editar Ver Marcadores Preferencias Ayuda
[edulcorante] > ~/E/P/D/practicaYoutube/AlfonsoGarcia > [L3 :1 !1] 13:29:12
make run-client-list
python3 client.py --Ice.Config=client.config
Enviando petición a
orchestrator -t -e 1.1
Petición de lista de archivos mandada...
[
{
 name = Poseidón
 hash = 9Meg1-02k5Q
},
{
 name = Ladilla Rusa - Macaulay Cylkin (VIDEOCLIP)
 hash = L38Z0YNJfCI
}]
[edulcorante] > ~/E/P/D/practicaYoutube/AlfonsoGarcia > [L3 :1 !1] 13:29:25
```

8

En caso de que queramos ver lo que imprimen los orchestrator, podemos usar la opción de *Retrieve stdout* de la interfaz gráfica. Pasa lo mismo con las *factory*.



*Ilustración 11 Retrieve stdout*