Proyecto ZeroMQ

8 de junio de 2015

1. Miembros del Grupo

- Juan Manuel Hidalgo Navarro
- Alexandra Morón Méndez
- Jose Carlos Solís Lojo
- Alejandro Rosado Perez

2. Introduccion

En nuestro proyecto con ZeroMQ vamos a llevar a cabo un sistema de transferencia de ficheros de forma remota, en el cual tendremos un cliente y un servidor, donde el servidor dará varias opciones al cliente y el cliente podra enviar o recibir ficheros entre muchas otras cosas.

3. Idea del Proyecto

Como bien hemos indicado, se realizará un proyecto con ZeroMQ que podra realizar un envio de fichero entre cliente y servidor. El servidor estará a la espera de cualquier peticion de un cliente, ya sea de almacenar un fichero, recibir un listado de los ficheros almacenados, obtener un fichero del servidor o incluso borrarlo.

4. Modo de Empleo

Para poder poner en marcha el sistema, debemos de ejecutar en una terminal el archivo **servidor.py** (con la orden "python servidor.py") y en otro terminal el fichero **cliente.py**, con la orden "python cliente.py.[en] caso de estar

ejecutandose ambos ficheros en la misma terminal (en localhost). Si el fichero **servidor.py** esta siendo ejecutado en otro ordenador, debemos de llamar a **cliente.py** pasandole como argumento la IP del servidor (ejemplo "python cliente.py 172.102.1.2"). Recordad que si no se le pone argumento, la IP a la que se conectara sera 172.0.0.1

Para resumir, si lo usamos en localhost, ejecutamos el servidor y despues el cliente con una llamada normal sin argumento, si se ejecutan en dispositivos distintos, la llamada al cliente debera tener el argumento de la IP donde se este ejecutando el servidor.

Una vez explicado esto, pasaremos a ver las opciones que nos da nuestro servicio.

5. Ordenes disponibles del sistema

Una vez que el servidor esta ejecutandose a la espera de una petición y ejecutamos un cliente, podemos elegir entre varias opciones:

- **?**: En primer lugar es el comando de ayuda que nos explica cuales son las opciones que podemos llevar a cabo y una breve explicacion.

```
Las ordenes disponibles son:
    list                    Listar archivos alojados en el
        servidor.
    del [archivo]           Borra un archivo del servidor.
    send [ruta_archivo]     Envia un archivo al servidor.
    get [archivo] [destino] Obtiene un archivo del servidor.
    exit                    Cierra la aplicaci n.
```

- **list**: Nos muesta un listado con todos los ficheros que se encuentran alojados en el servidor. Estaran alojados en una carpeta en el mismo directorio que nuestro archivo **servidor.py** llamada **files**

- **del [archivo]**: El cliente puede enviar la opcion de borrar uno de los archivos alojados en el servidor escribiendo la orden "del [nombre_del_archivo]"(sin corchetes). El servidor eliminará el fichero indicado de la carpeta **files**

- **send [ruta_archivo]**: Nuestro cliente envia un archivo indicando la ubicacion donde esta almacenado y el servidor guarda dicho archivo en la carpeta **files**. Ejemplo: "send ../Documentos/SD/Practica.py"Dicha orden enviaria el fichero Practica.py almacenado en la ruta indicada y el servidor lo recibiría guardandolo en la carpeta **files**.

- **get [archivo][destino]**: Obtiene el fichero indicado de la carpeta **files** del servidor y el cliente lo almacena en la ruta indicada. Ejemplo "get Practica.py ../Documento/SD/.El servidor enviaria el fichero Practica.py al cliente y este lo almacenaria en la ruta indicada.

- **exit**: Comando que simplemente cierra la aplicacion **cliente.py**