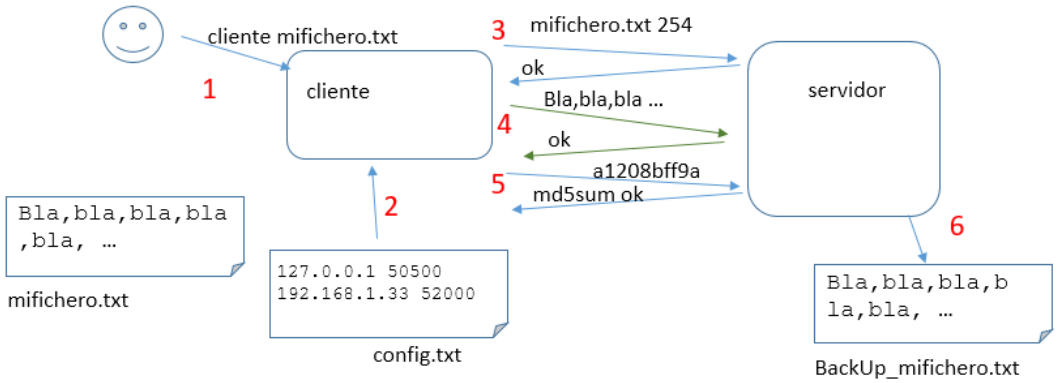
Trabajo final

Este trabajo intenta imitar el funcionamiento de un sistema de copia remota de ficheros. La aplicación distribuida consta de un cliente y un servidor. En la primera convocatoria sólo es necesario realizar el lado cliente, en la segunda convocatoria habrá que realizar ambos.

El mecanismo básico puede ser descrito como:

I. El usuario indica al programa cliente el nombre del fichero que desea copiar mediante línea de comandos al ejecutar el cliente. Este nombre de fichero será el único parámetro que se le pasa al cliente.
II. El programa cliente lee un fichero local de configuración denominado '*config.txt*' (que debe estar en el mismo directorio donde se ejecuta el cliente). El fichero de configuración está compuesto por líneas de texto. En cada línea se indica, separado por sólo un espacio en blanco, la dirección IP y un número de puerto (que se usará posteriormente tanto para TCP como UDP) donde se encuentra un proceso de tipo servidor. Se permite que haya servidores en ejecución de forma concurrente tanto en la misma máquina (tendrán la misma IP en cada línea pero puertos distintos) como en máquinas distintas. Como es lógico, no puede haber dos líneas exactamente iguales en el fichero.
III. El programa cliente envía un mensaje al servidor con el nombre del fichero introducido por el usuario en línea de comandos y su tamaño en bytes (concatenados por un solo espacio en blanco) *[2]* a través de UDP al puerto indicado en el fichero de configuración.
IV. Si el servidor responde '*ok*', entonces el cliente establece una conexión TCP con dicho servidor (al mismo número de puerto). A través de dicha conexión, el cliente envía el contenido del fichero indicado por el usuario. Se considera que el fichero a copiar sólo contiene caracteres, no es un fichero binario *[3]*. Si el mensaje recibido es '*transfer done*, el cliente cierra la conexión TCP.
V. Tras el cierre de la conexión TCP, el cliente calcula una huella del fichero transferido (hash o huella md5 que es un tipo de checksum) *[1]* y la envía al servidor a través de UDP al servidor por el mismo puerto inicial.
VI. El servidor calcula localmente la huella md5 del fichero recibido a través de la conexión TCP y la compara con la huella que ha recibido del cliente en el segundo mensaje de UDP. Si ambas coinciden, entonces graba el fichero recibido con el nombre que recibió en el primer mensaje UDP y envía la respuesta '*md5sum ok*.' al cliente a través de UDP.
VII. El cliente recibe la respuesta del servidor, si en el fichero de configuración tuviera más líneas (i.e. más de un servidor), volvería al paso III con el siguiente servidor. Cuando ya no tenga más servidores, cierra todos los sockets creados y termina el programa.

A continuación, podemos ver una ilustración del funcionamiento donde se muestra el funcionamiento de la aplicación distribuida. La comunicación entre cliente y servidor se ha dividido en tres fases: solicitud de envío, transferencia de fichero, comprobación md5. En este trabajo tienes que implementar el cliente.



La implementación

En un nivel más detallado, el cliente que implementes debe realizar los siguientes pasos (en este orden):

(a) Si el usuario escribe más de dos argumentos en línea de comandos (i.e. *sys.argv[0]* y *sys.argv[1]*) se muestra un error indicando '*Error. Uso: cliente [fichero]*' y se sale del programa.

(b) Se comprueba la existencia de al menos una línea en el fichero de configuración. En caso afirmativo, se comprueba la existencia del fichero referido en *argv[1]*). Si no existe el fichero de configuración o no tiene al menos una línea se indica el error *'Error: no hay servidores o fichero de configuración*'. Si el error es que no existe el fichero a transferir indicado por el cliente se indica el error '*Error. Fichero [filename] inexistente*' donde *[filename]* representa el nombre referido en *argv[1]*. En cualquiera de estos errores se sale del programa. No se comprueba que cada línea del fichero tiene el formato correcto, es decir, una dirección IP válida, un espacio y un puerto válido.

(c) Se envía al primer servidor de la lista un mensaje a través de un socket UDP. Dicho mensaje está compuesto por la concatenación (con un solo espacio en blanco como separador) del nombre del fichero a transferir y su tamaño (en bytes). A continuación se queda esperando la respuesta del servidor a través del mismo socket UDP. Si transcurridos 3 segundos no se recibe respuesta, se imprime el mensaje de error: '*Error: no hay respuesta por parte del servidor [IP] en el puerto [puerto]*' donde [*IP*] es la dirección IP del servidor y [*puerto*] es el puerto de escucha UDP del servidor. Si se produce este error pasamos al siguiente servidor del fichero de configuración. Si no hay este error pasamos al siguiente punto.

(d) La respuesta por UDP del servidor puede ser '*ok*' o '*no*'

   a. Si el servidor responde con '*ok*': entonces se abre el fichero, se lee su contenido (se supondrá que su contenido son caracteres y no es un fichero binario), y se envía dicho contenido a través de una conexión TCP con el servidor (en el mismo puerto que se indica en el fichero de configuración). El cliente debe realizar la conexión TCP y liberarla cuando termine. Una vez transferido el fichero por la conexión TCP, a

continuación se recibe por esta conexión TCP un mensaje del servidor que debería ser *'transfer done'* indicando que se ha recibido correctamente el contenido del fichero. Si pasados 10 segundos desde que inicia el envío por TCP, el cliente no recibe el mensaje *'transfer done'*, el cliente indica el error *'Error en la transferencia con servidor [IP]'*.

b. Si el servidor responde con '*no'*: entonces el cliente indica el error *'Error. El servidor [IP] no acepta el fichero'* donde *[IP]* es la dirección IP del servidor, y pasa al siguiente servidor.

(e) Finalmente, el cliente calcula una huella md5 del fichero enviado y la envía como *string* (método *hexhash()*) al servidor a través de su socket UDP inicial. A continuación queda a la espera de recibir por el mismo socket UDP un mensaje de confirmación:

a. Si el mensaje del servidor es '*md5sum ok*', el cliente imprimirá por pantalla: '*Copia de fichero en servidor [IP] correcta*', donde [*IP*] es la dirección IP del servidor con el que se ha mantenido el diálogo. En tal caso, el cliente ha finalizado correctamente la copia con este servidor y continuarían con el resto de servidores que le quedase en el fichero de configuración. El servidor escribirá en el disco en la misma carpeta donde se ejecutó el fichero recibido. Si el fichero ya existe se sobrescribe sin error.

b. Si el mensaje es '*md5sum error'*, el cliente imprimirá por pantalla: '*Error en la copia del fichero en el servidor [IP]. Se vuelve a intentar*', donde [*IP*] es la dirección IP del servidor con el que se ha mantenido el diálogo. En tal caso el cliente volvería a intentar transferir el fichero de nuevo con este servidor, desde el punto en el que recibió '*ok'* a su petición de copia. No se considera un número máximo de reintentos. Si los errores son continuos es porque hay algún problema en la programación.

c. Si transcurridos 10 segundos después de haberle enviado al servidor el hash: '*Error en la copia del fichero en el servidor [IP]. Finalizado el intento*', donde [*IP*] es la dirección IP del servidor con el que se ha mantenido el diálogo. En este caso, no se vuelve a intentar el envío con este servidor.

Consejo, desarrolla primero suponiendo que no hay errores ni excepciones. Después refina con excepciones. Finalmente, refina con errores.

Un ejemplo de ejecución del cliente y del servidor sería: (asegúrese de tener arrancado un servidor en el puerto indicado por el fichero de configuración).

ejecución del cliente	ejecución de un servidor sin errores
`%>python cliente.py ./fichero.txt`	`%>python server.pyc 8888  0`

Tratamiento de Errores

Errores que pueden ocurrir: (los que no se listen no se tienen por qué considerar y no se probarán)

- número de argumentos en línea de comandos erróneo.
- fichero *config.txt* no encontrado
- fichero a transferir no encontrado
- servidor no responde (timeout de 3 seg al primer mensaje UDP)
- transferencia denegada (respuesta al primer mensaje UDP es 'no')
- errores de sockets (p.ej. conexión no realizada, etc..)
- error en la transferencia (no se recibe '*transfer done*' del servidor en TCP con el timeout de 10 segundos).
- error en hash (respuesta final del servidor '*md5sum error*'), en cuyo caso hay que volver a transferir el fichero.
- error en hash (expira el temporizador de 10 segundos), en cuyo caso no hay que volver a transferir el fichero

Para poder comprobar los errores en los que el servidor puede incurrir, debe ejecutar el servidor con un último parámetro que indica el error que el servidor va a provocar. El significado del parámetro es:

- 0: Ejecución sin errores
- 1: Comprobación errónea del hash recibido y calculado
- 2: Fichero no aceptado por el servidor
- 3: El servidor espera 5 segundos antes de enviar '*ok*' o '*no*'
- 4: El servidor espera 15 segundos antes de enviar '*md5sum ok*' ó '*md5sum error*'
- 5: El servidor espera 15 segundos antes de enviar '*transfer done*'

No es posible simular más de un error de forma simultánea

Md5
[1]https://stackoverflow.com/questions/3431825/generating-an-md5-checksum-of-a-file
[1.5] https://docs.python.org/2/library/md5.html

Averiguar tamaño de un fichero
[2]https://stackoverflow.com/questions/2104080/how-to-check-file-size-in-python

Lectura de un fichero
[3] http://www.pitt.edu/~naraehan/python2/reading_writing_methods.html