PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2523 Sistemas Distribuidos (II/2017)

Tarea 2

Raimundo Herrera (rjherrera@uc.cl)

1 Solución

La solución implementada constó en leves modificaciones al archivo `node.py` y la creación de los archivos `download.py` y `upload.py`. Como estructura, se optó por tener, dentro de la carpeta `T2`, dos subdirectorios `files` y `source`. En el primero se guardarán los archivos compartidos en la red, y en el segundo el código del programa.
Como se señala en el `README.md` para ejecutar el nodo junto con el servidor de archivos, basta con escribir el siguiente comando estando en la carpeta `T2`:

```
python3 source/node.py
```

Análogamente, para las otras acciones, es posible realizar:

```
python3 source/download.py <file_to_download_name>
  python3 source/upload.py <file_to_upload_path>
```

1.1 `node.py`

En el archivo solo se modificó el puerto, incluyendo el asignado a mi (10009 y 11009 según correspondía) en los lugares pertinentes, además de agregar una linea de ejecución de un servidor HTTP con *python*. Esto se realizó por medio de la libreria `subprocess`, la cual permite ejecutar instrucciones de linea de comando en los programas. Así, al correr el archivo en cuestión, automáticamente se corre el servidor, y esto en la carpeta `files`, que se especifica al principio del archivo en la constante `FILES_FOLDER`.

1.2 `download.py`

En líneas generales el flujo del programa es muy simple, existe una función `download` que recibe una url, y por medio de la librería `requests` intenta descargar dicha url.
En la función principal `onResponse` se intenta descargar con la función anterior en cualquiera de las urls que se nos entregan (que corresponden a los lugares donde está guardado el archivo). En caso de no poder con una se intenta con la siguiente. En caso de si poder, se deja de preguntar por las siguientes, y se procede a registrarse como un nuevo servidor del archivo, de modo que ahora pueda ser descargado desde este nodo.

1.3 `upload.py`

Para esta parte simplemente se copia el archivo deseado a la carpeta definida anteriormente, esto es, la carpeta `files` donde están los *uploads*. Una vez copiado se procede a informar a la red que uno es quien tiene el archivo, de modo que quede registrado, y ahora uno sirva el archivo. Como hay un servidor HTTP de *python* corriendo en esa carpeta, automáticamente queda disponible el archivo.

2 ¿BitTorrent?

- ¿Que diferencias hay entre el esquema implementado y BitTorrent?

  En primer lugar, BitTorrent plantea un sistema *peer to peer* basado en trackers. Estos trackers tienen la información de quien posee los archivos. Funcionan como una especie de servidores centrales a los que se les consulta quiénes poseen un archivo para poder descargarlo desde ellos.

  En esa misma frase se sugiere otra diferencia, en nuestro esquema si bien obtenemos la información de quiénes tienen el archivo, únicamente lo descargamos de un nodo, y no de múltiples nodos, por pedazo, como se hace en BitTorrent. Esto hace que en nuestro sistema pueda haber un cuello de botella y que de cierto modo no se aproveche todo el potencial de la distribución, ya que solo se distribuye "el acceso" y no "la descarga".

  Otra diferencia es que en nuestro sistema hay potenciales fallas de seguridad y/o consistencia bastante grandes. Lo que se guarda en la red es el nombre del archivo, y nada garantiza que lo descargado sea lo mismo en todos los nodos. Es más, nada impide que 3 nodos compartan un archivo llamado 'pelicula.wmv' pero que los 3 sean diferentes. Esto porque a diferencia de BitTorrent, nosotros no *hasheamos* el contenido del archivo y se comparte únicamente el nodo. Por un lado el *hasheo* permite integridad y seguridad, ya que uno se asegura lo que baja, se pueden detectar cambios si el *hash* es lo suficientemente bueno, y por otro permite lo mencionado anteriormente distribuir la descarga.

- ¿Por qué razón no sería aconsejable almacenar los archivos directamente en la DHT?

  Si se guardan los archivos directamente se puede generar un *overhead* muy grande, ya que guardar archivos enteros de un tamaño arbitrario puede consumir la memoria de los nodos rápidamente. El mayor potencial de las tablas de hash a mi modo de ver es guardar información o metadata de los archivos y no los archivos completos. Si se guarda información referente al archivo y no el archivo mismo, la tabla de hash distribuida puede ser liviana y no presentar un problema para el nodo, si se tiene el archivo entero distribuido, como ya se dijo, se puede colapsar.

- ¿Qué ventajas/desventajas tiene utilizar una DHT vs utilizar trackers en este contexto?

  En primer lugar una DHT tiene la ventaja de que no se requiere de un ente externo para compartir archivos, sino que es posible compartirlos directamente de quien los posee. Esto también se logra con implementaciones de peer to peer, pero en ellas no se es 100% transparente ya que se incorporan los trackers, que hacen de servidores intermediarios. Si bien no tienen los archivos, hay que acceder a ellos para saber quien tiene cada archivo y de ese modo existe una dependencia extra de un nodo externo, perdiendo un poco la gracia de los sistemas peer to peer.

  Ese sistema de trackers trae una ventaja muy clara, que es que para saber quien tiene un archivo basta con preguntarle al tracker, que es conocido, en vez de tener que utilizar un algoritmo como el de Kademlia para averiguar quien posee cada archivo. Sin embargo, si un tracker no está disponible, se puede perder acceso a los archivos simplemente porque el encargado de informar de la ubicación de estos no está, aún cuando el archivo este siendo compartido.

  Por otro lado, en seguridad, una implementación como la nuestra presenta varias desventajas ya que no se asegura la integridad de un archivo, que no cambie. Ya que solo se verifica el nombre del archivo, mientras que, como ya se explicó, se utilizan en BitTorrent algoritmos de hasheo para tener una mayor integridad. También se pierde la capacidad de distribuir la descarga, ya que se descarga todo de un nodo y no de varios, como si se hace en BitTorrent. No obstante, con un poco de código adicional, esto se podría implementar en una DHT, siendo ellas virtudes no de BitTorrent en sí, sino que de la implementación, por lo tanto, sin necesidad de un tracker, y con las ventajas de un DHT.