Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2523 – Sistemas Distribuidos (II/2017)

Tarea 2

Entrega: 3 de Noviembre, 23:59

1 Objetivos

- Implementar el nodo de una red *peer to peer* utilizando la DHT (Distributed Hash Table) Kademlia para compartir archivos.
- Entender cómo funcionan los sistemas peer to peer en general.

2 Descripción

Los ayudantes han creado una librería en Python que deberán utilizar para implementar un nodo (simplificado) de una red *peer to peer* para compartir archivos.

La librería se encuentra en `http://github.com/vdrg/kademlia-iic2523`, donde además pueden encontrar más información y un ejemplo que les servirá para partir con su tarea.

2.1 Idea General

Una DHT o Distributed Hash Table es una estructura de datos distribuida que ofrece la misma interfaz que una Hash Table, es decir, las operaciones get(key) y set(key, value). La diferencia es que la información se almacena en distintos nodos de la red, y cualquier nodo puede setear/obtener los valores almacenados.

En *BitTorrent*, para empezar a compartir un archivo se debe computar un *"infoHash"* a partir de su contenido, que lo identificará en la red. Luego, en la DHT de BitTorrent se almacena la información del nodo (ip y puerto) usando el infoHash como llave (otra opción es usar un *tracker*, pero eso no nos interesa para esta tarea). De esta forma, si un usuario quiere descargar el archivo, simplemente busca en la DHT las ip's y puertos de los nodos que están compartiéndolo, y a la vez publica en la DHT su ip y puerto. El nodo se conectará a algunos de los nodos que encontró y, siguiendo cierto protocolo, descarga/comparte el archivo por partes.

Para esta tarea tendrán que implementar una versión ultra simplificada de éste esquema. La diferencia principal es que no se deben preocupar de implementar ni la DHT ni el protocolo para descargar archivos. Para la DHT utilizarán la librería mencionada anteriormente, y para compartir sus archivos simplemente deberán servirlos con un servidor http.

2.2 Funcionamiento

Deberán modificar el ejemplo que se encuentra en el repositorio de la librería. El ejemplo contiene 4 archivos:

- **server.py**: Este nodo estará corriendo en el clúster y será el nodo de "entrada" a la red (no deben modificarlo, está sólo como ejemplo).
- **node.py**: El programa principal, que utiliza la librería para obtener y almacenar valores en la DHT. Al leer el código, se darán cuenta que se ejecuta la operación *server.bootstrap([("192.168.1.106", 8468)])*, cuyo objetivo es conectarse con el nodo corriendo en el clúster (obviamente, esta operación será exitosa sólo si ejecutan el programa dentro del clúster). Además, recibe mensajes a través de sockets unix de los siguientes programas:
- **get.py**: Este programa recibe como argumento la llave que quieren buscar en la DHT. Para hacer esto, simplemente le envía al nodo el string "get key", y recibe como respuesta los valores encontrados.
- **set.py**: Este programa recibe como argumentos la llave y el valor que quieren almacenar en la DHT. Al igual que el programa anterior, simplemente le envía al nodo el string "set key value".

Ustedes tendrán que modificar **node.py** y crear 2 programas (modificando **get.py** y **set.py**):

- **upload.py**: este programa recibe como argumento el path a un archivo local. Este archivo deberá ser servido en la URL que ustedes quieran (por ejemplo, *localhost:PORT/filename*) (todos servirán sus archivos en dentro del cluster, información sobre los puertos y nodos más abajo). Para que sus compañeros puedan obtener la URL dado el nombre del archivo, deberán almacenarla en la DHT, siendo la llave el nombre del archivo.
- **download.py**: este programa recibe como argumento el nombre de un archivo, y deberá hacer un request a su nodo para obtener las URLs correspondientes al nombre del archivo. Luego, deberá descargarlo desde alguna de las URLs obtenidas (si alguna no está disponible, deben intentarlo con las otras). Una vez descargado, deben empezar a servir el archivo y almacenar en la DHT la URL respectiva.

Pueden servir los archivos como ustedes prefieran.

2.3 Puertos y Nodos

Por simplicidad, su programa deberá correr y servir los archivos dentro del clúster. Dado que varios podrían estar probando sus nodos en la misma máquina del clúster, cada uno utilizará puertos distintos, tanto para el nodo de Kademlia como para el servidor http.

El nodo de ejemplo recibe un puerto como argumento, pero ustedes deben hardcodear su puerto asignado para la DHT. El puerto que deben utilizar **para el nodo de la DHT** será $10000 + NumeroDeLista$ (es decir, el primero de la lista debe utilizar el puerto 10001, el segundo el 10002, etc). Sus números de lista los pueden encontrar en el excel del curso. Además, **para el servidor http** deberán usar el puerto $11000 + NumeroDeLista$.

El servidor de entrada estará corriendo en makemake. Ustedes pueden correr sus nodos y servidores en makemake, titan, tripio, caleuche o trauco.

2.4 Ayudita

La tarea no es tan compleja una vez que entienden cómo manipular la librería. No se compliquen con la parte de servir archivos: una buena idea es simplemente tener un directorio 'uploads' que es servido permanentemente, y al compartir/descargar un archivo simplemente lo copian a ese directorio.

Para empezar a servir un directorio fácilmente, pueden ejecutar *python -m SimpleHTTPServer PORT* dentro de éste.

3 Informe

Deben entregar un informe que contenga lo siguiente:

- Descripción breve de la solución implementada.
- Respuestas para las siguentes preguntas:
  - ¿Que diferencias hay entre el esquema implementado y BitTorrent?
  - ¿Por qué razón no sería aconsejable almacenar los archivos directamente en la DHT?
  - ¿Qué ventajas/desventajas tiene utilizar una DHT vs utilizar *trackers* en este contexto?

4 Restricciones

La tarea debe ser programada en Python 3.

5 Evaluación

- 10%. Aspectos formales: entrega correcta, instrucciones de ejecución apropiadas detalladas en un archivo `README`, en formato *txt* o *markdown*.
- 10%. Calidad del código (sean ordenados e incluyan comentarios).
- 20%. Descarga de archivos (sin servirlos una vez descargados).
- 25%. Subida de archivos y registro de las URLs en la DHT.
- 15%. Servir un archivo una vez descargado, y registrar la URL en la DHT.
- 20%. Informe.

6 Entrega

Independientemente de la plataforma en que haya ejecutado sus experimentos, la entrega debe realizarse en una carpeta de nombre **exactamente** `T2` en su directorio *home*.

Deben incluir un archivo con instrucciones para correr sus programas. No es necesario que su nodo se ejecute con python directamente (el código debe ser en python, pero si quieren pueden utilizar un script bash o algo por el estilo para correrlo).

Contacto

- Vicente Rafael Dragicevic (vrdragicevic@uc.cl)
- Pilar Ignacia Jadue (pijadue@uc.cl)
- Francesca Lucchini (flucchini@uc.cl)

7 Integridad académica

Los alumnos de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile deben mantener un comportamiento acorde a la Declaración de Principios de la Universidad. En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un Procedimiento Sumario. Es responsabilidad de cada alumno conocer y respetar el documento sobre Integridad Académica publicado por la Dirección de Docencia de la Escuela de Ingeniería.

Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno, sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno copia un trabajo, obtendrá nota final 1.1 en el curso y se solicitará a la Dirección de Docencia de la Escuela de Ingeniería que no le permita retirar el curso de la carga académica semestral. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona.

Obviamente, está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la referencia correspondiente.

Lo anterior se entiende como complemento al Reglamento del Alumno de la Pontificia Universidad Católica de Chile. Por ello, es posible pedir a la Universidad la aplicación de sanciones adicionales especificadas en dicho reglamento.