TP3: Desarrollo de un sistema distribuido simple

Sistemas Distribuidos - Ingeniería Informática, Universidad Nacional de Mar del Plata

Versión: 1.1.0
Fecha: 28/09/2021

Revisión	Cambios realizados	Fecha
1.0.0	Creación del documento y estructura de secciones principales. Redacción de los primeros ejercicios.	10/09/2021
1.0.1	Actualización diagrama de arquitectura.	13/09/2021
1.1.0	Se agregó el Apéndice 1: Implementación sugerida de DHT.	28/09/2021

Tabla de Contenidos

Introducción

Este trabajo práctico propone desarrollar un sistema distribuido utilizando algunas herramientas y conceptos tratados en la materia. El producto esperado es un sistema de intercambio de archivos, inspirado en software como BitTorrent y similares.

IMPORTANTE: Tanto la arquitectura como los requerimientos son una guía para que los alumnos desarrollen un sistema distribuido. El objetivo es que los alumnos tomen decisiones de diseño y realicen una implementación que les permita enfrentarse con algunos de los desafíos que surgen a la hora de desarrollar este tipo de sistemas. Por lo tanto, los requerimientos y la arquitectura pueden estar sujetos a cambios propuestos por los mismos alumnos y aprobados por la cátedra.

Arquitectura

El sistema cuenta con los siguientes tipos de componentes:

C-1. Grupo de nodos tipo tracker
C-2. Grupo de nodos tipo par
C-3. Servidor web
C-4. Cliente web

Se denota como C-1.x y C-2.x a los componentes de tipo C-1 y C-2 respectivamente, donde x es un número que identifica unívocamente a una instancia de dicho tipo. Por ejemplo, los nodos tipo tracker serán identificados como C-1.1, C-1.2, C-1.3, etc, y los nodos tipo par serán identificados como C-2.1, C-2.2, C-2.3, etc.

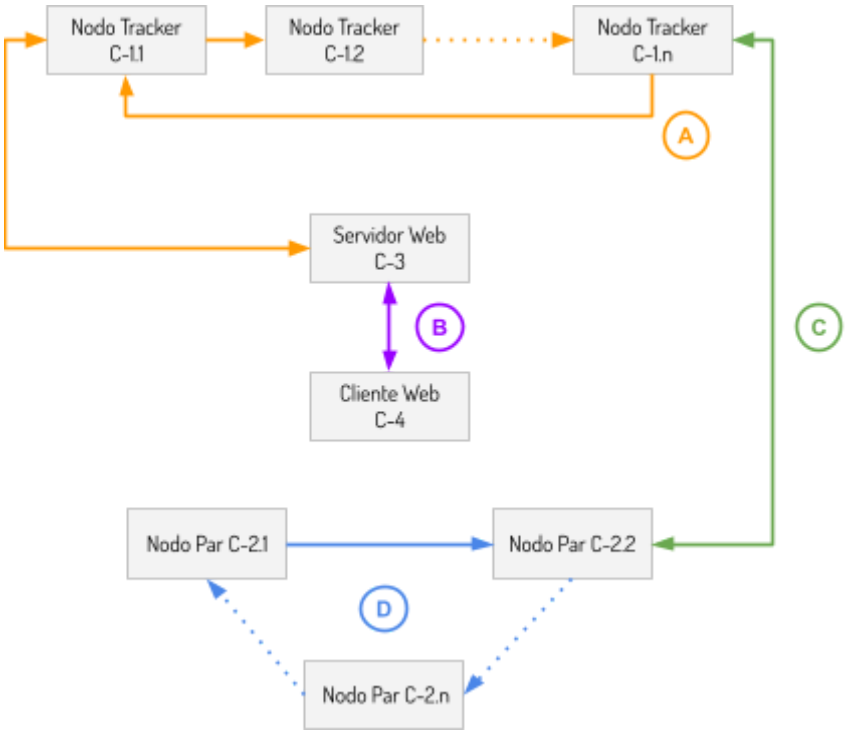
El grupo de nodos tracker (C-1) almacena mediante DHT (Distributed Hash Table) la información de los archivos disponibles para intercambio. Básicamente, cada nodo guarda en memoria una estructura tipo diccionario, donde la clave es un hash y su valores son el nombre del archivo, el tamaño (en bytes redondeado), y que pares (sockets TCP) lo ofrecen. Los nodos tienen identificadores únicos y están vinculados mediante una lista circular doblemente enlazada, permitiendo a cada nodo conocer a sus 2 vecinos. Los hash deben estar distribuidos en los nodos en orden alfabético para que pueda realizarse fácilmente su búsqueda. Cada nodo debe intentar almacenar la misma cantidad de archivos, que surge de la operación aritmética archivos/trackers (aunque no hace falta redistribuir archivos si ya fueron asignados a un nodo). Para realizar operaciones sobre el DHT (por ejemplo la búsqueda de un archivo a partir de su hash), se pasan solicitudes los nodos unos a otros, recorriendo la lista circular en 1 sentido, hasta que la respuesta vuelve al origen.

El servidor web (C-3) y su respectivo cliente (C-4) ofrecen por un lado un servicio donde dar de alta nuevos archivos en los trackers (C-1) y por otro lado un escaneo de los archivos que tienen registrados los trackers (C-1). De esta forma, un usuario puede ingresar a una interfaz gráfica y enviar archivos o ver qué archivos están disponibles para descargar, y

bajar su información para que sea ejecutada por un nodo tipo par (C-2). Esta información se envía o descarga en un archivo con la extensión “.torrente”.

Un nodo par (C-2) ofrece archivos para que otros pares descarguen, y a su vez se conecta a otros pares para descargar archivos, a partir del contenido de los archivos “.torrente”.

El diagrama a continuación muestra las interacciones de los componentes y los protocolos de red que utilizan.



Interfaces:

- **A** - Protocolo UDP. Arquitectura P2P. Mensajes principales:
    - Search: un nodo solicita la búsqueda de un archivo
    - Scan: solicita un escaneo de archivos del vecino o de toda la red
    - Found: devuelve ninguno, uno o más archivos encontrados
    - Store: se solicita almacenar un nuevo archivo
    - Count: cuenta la cantidad de nodos y archivos en la red
    - Join: un nodo quiere ingresar a la red
    - Leave: un nodo se va de la red
    - Node Missing: un nodo no se encuentra
    - Heartbeat: informe periódico para avisar que el nodo sigue funcionando
- **B** - Protocolo HTTP. Arquitectura Cliente-Servidor. Mensajes principales:
    - Alta de archivos
    - Listar archivos
    - Solicitud de descarga de “.torrente”
- **C** - Protocolo UDP. Cliente-Servidor. Mensajes principales:

    - Solicitud de archivo “.torrente”
- **D** - Protocolo TCP. Arquitectura P2P. Mensajes principales:
    - Solicitud de descarga de archivo

Requerimientos

Los siguientes requerimientos actúan como complemento de la arquitectura descrita. Los mismos pueden sufrir modificaciones o surgir nuevos a medida que avance el desarrollo.

Requerimientos funcionales

F-1. El cliente web debe permitir la carga de nuevos archivos a los trackers mediante el completado de un formulario (nombre de archivo, tamaño, y socket de par que lo contiene).
F-2. El cliente web debe permitir listar todos los archivos disponibles en los trackers (nombre y tamaño).
F-3. El cliente web debe permitir la descarga de archivos “.torrente” (a partir de la lista mencionada anteriormente).
F-4. Los pares deben consultar al tracker, que se encuentra en el archivo “.torrente”, la lista de pares que contienen el archivo a descargar.
F-5. Los pares deben descargar archivos desde otros pares, a partir de la información encontrada en el archivo “.torrente” (consultando primero al tracker).
F-6. Los pares y los trackers leen su configuración inicial (cantidad de nodos, direcciones, etc) a partir de un archivo JSON de configuración.
F-7. Los trackers deben poder aceptar nuevos nodos de manera dinámica (es decir, una vez comenzado el sistema).
F-8. Los trackers deben permitir que un nodo se apague manualmente y redistribuir su contenido en los nodos vecinos.
F-9. Los trackers deben implementar un heartbeat con sus vecinos, y en el caso de que este se interrumpa, dar por caído al vecino.
F-10. Los trackers deben ser tolerantes ante la caída de un nodo, manteniendo redundancia de información con sus vecinos (es decir, deben tener una copia de la información de sus vecinos, y en el caso de que se caiga deben volver a generar los enlaces con los nuevos vecinos y volver a copiar la información).

Requerimientos no funcionales

NF-1. Los hash se deben realizar utilizando el algoritmo SHA-1 pasando como argumento la concatenación del nombre del archivo y el tamaño en ASCII (en bytes) redondeado.
NF-2. Los pares deben estar ejecutándose de manera permanente para permitir la descarga de archivos (tanto los que desea obtener como los que ofrece).
NF-3. Los pares deben ofrecer una interfaz de línea de comando (tipo prompt).
NF-4. El archivo “.torrente” contendrá la siguiente información en formato JSON:

    - Nombre de archivo
    - Tamaño en bytes
    - Dirección socket del tracker

NF-5. Toda comunicación, a excepción de la descarga del archivo entre pares, deberá utilizar el protocolo JSON:API (https://jsonapi.org).

NF-6. Los sistemas desarrollados por los grupos deben ser interoperables.

Metodología de trabajo y evaluación

Cada grupo deberá presentar un plan de hitos de desarrollo (con fecha y alcance de cada hito) para antes de la fecha establecida en este documento.

Cada grupo deberá utilizar un repositorio Git propio (público o privado brindando acceso a docentes) en el que deberán ir subiendo los avances. Es importante que cada miembro del equipo sea colaborador y participe activamente del repositorio ya que será parte de la evaluación.

Cada grupo deberá presentar, durante las clases de práctica, el avance y cumplimiento de los hitos del plan, en aproximadamente 15 minutos. Se dará espacio también para discutir y coordinar con el resto de los grupos el diseño de las interfaces para que sean interoperables.

Cada grupo debe llevar un documento de minutas independiente, con los puntos que consideren importantes que se discutieron en las clases con respecto al trabajo. Además, las interfaces se deberán ir definiendo en conjunto en un mismo documento colaborativo. Tanto las minutas como el documento de interfaces se deberán ir guardando en la siguiente carpeta: https://drive.google.com/drive/folders/1oW-AaJNTtz0pyW20TDyS4rkQwiFe0PSp

La implementación y defensa de este trabajo será evaluada, y es condición necesaria su aprobación para aprobar la cursada. La implementación puede ser grupal, pero la defensa será oral e individual. Consistirá en responder preguntas conceptuales y técnicas relacionadas con el desarrollo del mismo.

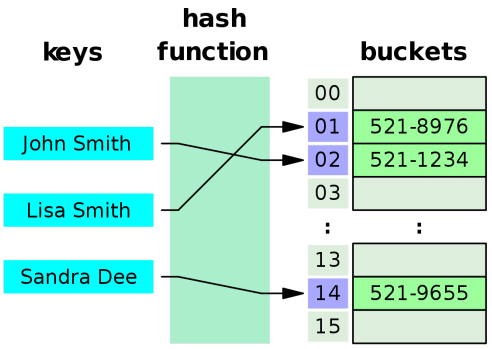
Para la promoción, se requiere realizar aquellas características marcadas en rojo (no requeridas para habilitar).

Los grupos deberán respetar el siguiente “Calendario de entregas”:

- 13/09: Entrega de lista de equipos
- 27/09: Fecha límite de entrega de plan de hitos y repositorio Git
- 15/11: Entrega parcial de trabajo (en Git)
- 30/11: Entrega final (en Git) y Defensa oral (demo y presentación)

Apéndice 1: Implementación sugerida de DHT

Una tabla hash es una estructura tipo diccionario (clave-valor) que permite una búsqueda eficiente de sus elementos, generalmente mapeando claves a índices de vectores, utilizando una función hash. Por ejemplo, si en una tabla hash que almacena libros en PDF, buscamos el nombre de un libro determinado, este sería convertido a un número mediante una función hash y ese número representaría un índice en un arreglo que contiene dicho libro en formato PDF. De esta forma, la eficiencia se plasma en el acceso al elemento utilizando un índice de un vector que es más rápido que realizar una búsqueda por comparación de texto.



Al aplicar una función hash sobre un elemento, pueden ocurrir colisiones, lo que implicaría que más de un elemento tenga un mismo índice. En este caso, se podría resolver teniendo una nueva estructura dentro de cada índice que permita la búsqueda de todos los elementos que coinciden en el hash.

Las DHT son tablas hash con la particularidad que están distribuidas en más de un nodo, en una arquitectura P2P. Los nodos están enlazados entre sí y mientras más enlaces tenga cada nodo, más rápidas serán las operaciones (búsqueda y escritura) pero más complejo será el mantenimiento de la estructura. La cantidad de enlaces que tenga cada nodo determina la topología de la DHT.

Para la implementación se sugiere una lista circular, doblemente enlazada, donde las operaciones recorran un sólo sentido (a no ser que sea necesario cambiar el sentido porque por ejemplo un nodo no responde). Para el indexado, se sugiere utilizar los primeros 2 dígitos del hash SHA-1 (por ejemplo, si el hash fuera “3a0cb0fc0983acdb687b4685154a75bd6eccd63a”, tomar “3a” que es el índice 58 en decimal). Esto va a dar un total de 255 índices disponibles. Como pueden ocurrir colisiones en 255 índices, dentro de cada elemento del arreglo se guardan todos las claves que colisionan en esos 2 dígitos (por lo que se debe realizar una nueva búsqueda en cada elemento). Se puede asumir (o no, a criterio del alumno) que no hay colisiones en la cadena de hash completa.