Proyecto Final Redes

Por: John Sanabria - john.
sanabria@correounivalle.edu.co

Definición

Construir una aplicación similar a BitTorrent (P2P) para la transferencia de archivos

- Basada en TCP
- La aplicación será capaz de descargar varios “chunks” de forma simultánea

Definición del proyecto

- Este proyecto se basa en el protocolo para transferencia de archivos BitTorrent P2P
- En un ambiente cliente/servidor, el cliente sabe donde acceder al archivo
- En un ambiente P2P la ubicación de un archivo no es claro. El archivo puede no estar en una sola parte sino en muchas

Conceptos introducidos por BitTorrent

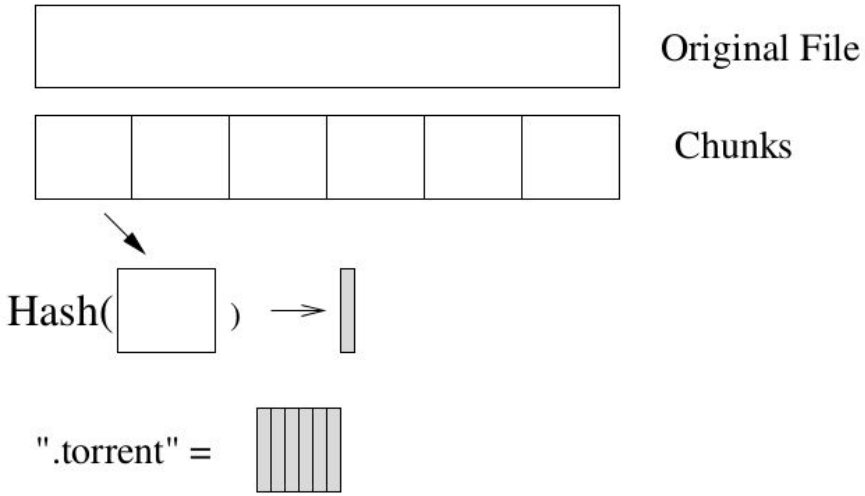
- Chunk, unidad básica de transferencia y almacenamiento. Su tamaño es de 512 KB y cada archivo deberá particionarse con esa unidad mínima
- Los chunks pueden ser descargados de forma independiente y sin “ningún” orden particular
- Al descargar todos los chunks, el archivo podrá ser reconstruido

Conceptos introducidos por BitTorrent

- Tracker, es una “entidad” que conoce quien tiene que “chunks” de un archivo
- Un “cliente” descarga un .torrent
  - .torrent lista la información sobre cada “chunk” de un archivo
- Un chunk es identificado por su hash criptográfico. Para cada chunk descargado se debe calcular su hash criptográfico

Operación de BitTorrent

- Un chunk es identificado por su hash criptográfico. Para cada chunk descargado se debe calcular su hash criptográfico



Operación de BitTorrent

- Un peer solicitando un chunk particular recibirá del tracker una lista de peers que contienen el chunk
- BitTorrent utiliza la heurística “rarest-chunk-first” para determinar que chunk se descargará primero
- ~~*El peer puede descargar 4 chunks al tiempo*~~

Como debe operar su “BitTorrent”

- No hay un tracker. Se utiliza un principio de inundación para determinar que peer tiene que chunk
- Cada nodo conoce la identidad de los demás nodos en la red
- No se debe implementar un mecanismo basado en incentivo para premiar a los buenos “uploaders” y castigar a los malos

Algunos archivos importantes

- Un archivo de nodos. Los nodos o peers hacen referencia a aquellos equipos que pertenecen a la red P2P

```
1  192.168.28.7     4500
2  192.168.28.9     6700
```

- El archivo asigna un número decimal al peer, indica su número IP y el número de puerto donde corre la aplicación
- Todos los peers deberían tener el mismo archivo

Algunos archivos importantes

- Un archivo de chunks que define los chunks que constituyen un archivo

```
File: nombre_archivo
Chunks:
id_0    chunk_hash
id_1    chunk_hash
```

- El formato indica un nombre de archivo y los chunks que constituyen este archivo
- id es un número decimal e indica la posición del chunk en el archivo

Algunos archivos importantes

- Un archivo que indica que chunks tiene un nodo o peer

```
id_k        hash_chunk
id_k+1      hash_chunk
id_m        hash_chunk
```

- Este archivo es dinámico en cuanto a que a medida que un peer obtiene un nuevo chunk su lista de chunks conocidos crece

Descripción de operación de su proyecto

- Cuando un cliente requiere un archivo, este envía un mensaje a todos los peers que se define en el archivo de nodos
  - Deberá enviar un comando que diga “`GET <filename>`”
- Todos aquellos nodos que contengan el `<filename>` le enviarán al cliente su lista de `chunk_hash`. El mensaje deberá tener el comando “`HAS <chunk-list>`”
  - NOTA: Un cliente puede solicitar mas de un archivo a la vez
- El cliente usará la heurística `rarest-chunk-first` para comenzar a hacer la descarga

Descripción de operación de su proyecto

- El cliente usará la heurística `rarest-chunk-first` para comenzar a hacer la descarga
  - El cliente enviará un mensaje “`SENDME chunk_hash`” al peer que lo posea
  - Una vez un chunk este disponible, si otro peer solicita el archivo que se esta descargando, este cliente deberá anunciar que él tiene chunks por compartir
- Una vez el archivo haya sido totalmente descargado el aplicativo deberá imprimir un mensaje indicando que el archivo esta listo de forma local

Acerca de la implementación

- Debe escribirse en C. No se permite C++ o STL
- Su código debe correr bajo Linux
- Para programar los sockets ud. debe usar las librerías **standard** para programación de sockets
- Debe proveer documentación respecto a los diferentes archivos de su proyecto. Los archivos deben también estar documentados

Acerca de la implementación

- Debe guardar sus proyectos en un repositorio en bitbucket
- Su aplicación debe soportar un flag (`--debug`) que permita ver en detalle la interacción que se sucede cuando se descarga un archivo

Constitución de los grupos

- El máximo número de miembros por grupo es 3 y mínimo 2

Entregas

- Semanalmente, o no, se pondrán tareas en esta presentación
- Cada tarea planteada deberá ser implementada por su grupo. Las tareas planteadas serán evaluadas cada semana
- Cuando se adicione una nueva tarea o

Tareas

- Noviembre 4
  - Nombre de los miembros de los grupos
  - Dado un archivo particionarlo en 512 KB, en C
  - Dado un archivo (chunk) calcularle su SHA-1, en C