PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA
CAMPUS SANTO TOMÁS DE AQUINO

Asignatura:

Sistemas Operativos II
ISC-365- T- 001

Profesor:

Roberto Abreu

Tarea
Reporte Cliente BitTorrent

Estudiantes:

Aydee Rodríguez 2014-5761
Paola Nieves 2014-6304

Fecha de entrega:

31 de marzo de 2019

Arquitectura

Para realizar la implementación del cliente de BitTorrent se realizaron los siguientes pasos:

1. Parsear el .torrent file

El primer paso que debe de realizar un cliente es encontrar qué archivo se quiere descargar y de donde lo descargarlo. Esta información está almacenada en el .torrent file conocido también como meta-info.En este archivo hay varias propiedades que se necesitan para implementar un cliente.

Entre estas propiedades está:

- El nombre del archivo para descargar.
- El tamaño del archivo a descargar.
- El URL al tracker.

Todas estas propiedades están almacenadas en un formato binario,llamado Bencoding,por esta razón necesita ser decodificado.

2. Conectarse al tracker

El segundo paso después de decodificar el .torrent file y se tenga una representación en python de la data ,tenemos que conseguir la lista de peers a conectarse, y es aquí que necesitamos al tracker ,que es un servidor central que da seguimiento de los peers disponibles para un torrent determinado. Este no contiene información ninguna información del torrent solo cuales peers se puede conectarse y sus estadísticas.

Para construir el request,se necesita el announce del .torrent file que es un URL http al tracker, utilizando los siguientes parámetros:

- **info_hash:** urlencoded de 20-byte SHA1 hash , valor del info key del torrent file.
- **peer_id:** un id único generado para el cliente.
- **uploaded:** el número total de bytes cargados.
- **downloaded:**el número total de bytes descargado.
- **left:**el número de bytes que quedan por descargar para el cliente.
- **port:** el puerto TCP que el cliente escucha.

Un request exitoso utilizando el announce para llegar al tracker , nos devuelve una respuesta con una lista de peers a conectarse,es posible no estén todos los peers disponibles del swarm,

solo los peers a los que el tracker asignó al cliente para conectarse. Una llamada subsiguiente al tracker podría resultar en otra lista de peers.

De la respuesta del tracker se obtiene dos propiedades importantes:

interval: el intervalo en segundos hasta que el cliente realice una nueva llamada announce al tracker.

peers: la lista de peers es una cadena binaria con una longitud de múltiplo de 6 bytes. Donde cada peer consiste en una dirección IP de 4 bytes y un número de puerto de 2 bytes.Se necesita parsear esta respuesta para obtener la dirección IP y el puerto.

4. Conectarse a los peers

Las conexiones entre peers se realizan a través de TCP a la IP y el puerto del host correspondiente, esta comunicación se hace a través de sockets.Después de que se realiza la conexión a los peers, estos peers comenzarán a intercambiar mensaje, el primer mensaje es el ***Handshake*** que inicia el cliente.

Las conexiones de los clientes comienzan como "choked" y "no interesadas". Un cliente debe mantener la información del estado para cada conexión que tenga con un peer remoto:

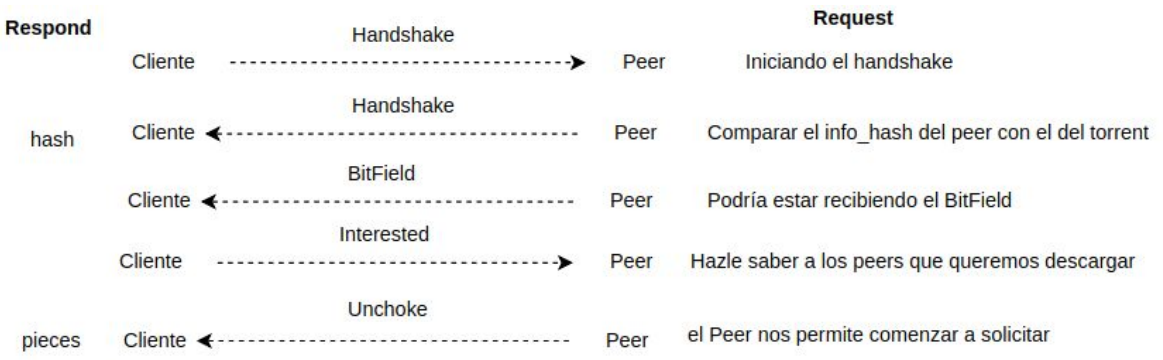
- **choked**: eso significa que al cliente no se le permite realizar un request de piezas de un peer remoto.
- **interested**: si el peer está interesado en algo que el cliente tiene para ofrecer. Esto ocurre cuando el cliente realiza unchoked a los peers.

El mensaje de Handshake contiene dos campos importantes:

- peer_id: el id único de los peers.
- info_hash: El valor hash SHA1.

Si el info_hash no coincide con el torrent que estamos a punto de descargar, se cierra la conexión.Inmediatamente después del Handshake, el peer puede enviar un mensaje de BitField. El mensaje de BitField sirve para informar al cliente sobre qué piezas tiene el peer.

Al final quedaría una secuencia de mensajes que es la que queremos replicar para realizar una conexión exitosa con los peers.Como se muestra en la siguiente imagen:



Tan pronto como el cliente se encuentre en un estado de unchoked, comenzará a solicitar piezas del peer conectado.

Tipos de mensajes

Los mensajes enviados después del Handshake esta estructurados de la siguiente manera:

[message length as an integer] [single byte describing message type] [payload]

Entre los tipos de mensajes se encuentran:

- Los mensajes de Keep Alive se envían con intervalos regulares, y son simplemente un mensaje con longitud 0, y sin tipo ni payload.

- Los tipos de mensajes 0, 1, 2, 3 que son choke, unchoke, interested y not interested respectivamente. Todos ellos tienen longitud 1 y sin payload. Estos mensajes simplemente describen cambios de un estado.

- El mensaje have es de tipo 4 que tiene una longitud 5, y un payload que es un solo entero, que es índice de la pieza del archivo que el peer tiene descargado.

- El mensaje bitfield de tipo 5,este mensaje solo se envía directamente después del handshake. Contiene una representación en bits de las piezas que tiene el peer. Un bit establecido en 1 representa la pieza. Los peers que no tienen piezas pueden omitir el envío de este mensaje.

- El mensaje de tipo 6 es request, el payload ,consiste en tres enteros, índice de la pieza, begin y longitud. El índice de la pieza decide cual pieza el cliente desea descargar, begin da el byte offset dentro de la pieza, y la longitud es la cantidad de bytes que el cliente desea descargar. La longitud suele ser una potencia de dos.

- El mensaje block es de tipo 7,este mensaje va de seguido de un request y el mensaje de tipo 8 es cancel y es utilizado para cancelar un request.

Los peers deben actualizar su estado continuamente su estado de interés a sus peers vecinos, para que el cliente sepan qué peers comenzarán a descargar cuando no estén unchoked.

5. Client Download

Entonces si ya sabemos que un peer tiene una pieza determinada, podemos enviar un mensaje de **request** pidiéndole al peer que nos envíe datos para la pieza especificada. Si el peer responde, nos enviará un mensaje con la pieza correspondiente donde el payload del mensaje es la información.

6. Client Upload

Para poder subir piezas y compartirlas con otros peers en vez de esperar por los mensajes , mandamos nosotros los mensajes, estando escuchando en un a conexión hasta que un peer quiera conectarse, se verifica el Handshake y enviamos un mensaje BitField para que los otros peers sepan que piezas del archivo tiene el cliente, luego si es posible mandamos un mensaje unchoke para que los demás peer tengan permiso a acceder a las piezas que tiene el cliente, luego si un peer responde ,recibimos un mensaje request del peer , pidiendo que le envíe una parte file, si se tiene esa pieza se envia al peer y vuelve se repite esto pasos, para seguir subiendo las piezas del archivo que tengamos.

Resultados

Download

5999	1131.8556062…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x84,Begin:0x0,Len:0x4000
6000	1131.8561233…	10.0.3.15	62.210.167.128	BitTor…	73	Request, Piece (Idx:0x85,Begin:0x0,Len:0x4000)
6014	1132.1580358…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x85,Begin:0x0,Len:0x4000
6016	1132.1590469…	10.0.3.15	62.210.167.128	BitTor…	73	Request, Piece (Idx:0x86,Begin:0x0,Len:0x4000)
6021	1132.4514504…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x86,Begin:0x0,Len:0x4000
6022	1132.4520926…	10.0.3.15	62.210.167.128	BitTor…	73	Request, Piece (Idx:0x87,Begin:0x0,Len:0x4000)
6029	1132.7418830…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x87,Begin:0x0,Len:0x4000
6030	1132.7426461…	10.0.3.15	62.210.167.128	BitTor…	73	Request, Piece (Idx:0x88,Begin:0x0,Len:0x4000)
6037	1133.0295611…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x88,Begin:0x0,Len:0x4000
6038	1133.0305507…	10.0.3.15	62.210.167.128	BitTor…	73	Request, Piece (Idx:0x89,Begin:0x0,Len:0x4000)
6046	1133.3243650…	62.210.167.128	10.0.3.15	BitTor…	1040	Piece, Idx:0x89,Begin:0x0,Len:0x4000
6269	1173.6421785…	62.210.167.128	10.0.3.15	BitTor…	62	BitTorrent
6581	1293.6196344…	62.210.167.128	10.0.3.15	BitTor…	62	BitTorrent

Upload

3087	761.405435011	127.0.0.1	127.0.0.1	BitTor…	136	Handshake
3089	761.405673787	127.0.0.1	127.0.0.1	BitTor…	136	Handshake
3091	761.407791588	127.0.0.1	127.0.0.1	BitTor…	133	Bitfield, Len:0x3c
3093	761.410991993	127.0.0.1	127.0.0.1	BitTor…	73	Interested
3094	761.411040048	127.0.0.1	127.0.0.1	BitTor…	73	Unchoke
3095	761.411188309	127.0.0.1	127.0.0.1	BitTor…	85	Request, Piece (Idx:0x0,Begin:0x0,Len:0x4000)
3254	817.653636663	127.0.0.1	127.0.0.1	BitTor…	136	Handshake
3256	817.654021520	127.0.0.1	127.0.0.1	BitTor…	136	Handshake
3258	817.656998162	127.0.0.1	127.0.0.1	BitTor…	133	Bitfield, Len:0x3c
3260	817.658919715	127.0.0.1	127.0.0.1	BitTor…	73	Interested
3261	817.658952069	127.0.0.1	127.0.0.1	BitTor…	73	Unchoke
3262	817.659024630	127.0.0.1	127.0.0.1	BitTor…	85	Request, Piece (Idx:0x0,Begin:0x0,Len:0x4000)

Referencias

Additional information on the BitTorrent Protocol http://wiki.theory.org/BitTorrentSpecification.

Wiki.theory.org. (2019). BitTorrentSpecification - Theory.org Wiki. [online] Available at: https://wiki.theory.org/index.php/BitTorrentSpecification.