Explicación de Protocolo Peer-Tracker UDP[1]

El objetivo de la comunicación Peer-Tracker es la generación del swarm, y que el Peer tenga la lista de otros Peers del swarm para poder intercambiar el contenido. La variante UDP del protocolo Peer-Tracker tiene la ventaja de reducir el volumen de información intercambiada hasta es un 50% respecto al protocolo basado en TCP.

El primer paso para comenzar la comunicación es la obtención de un ID de conexión (connection_id). Este ID se utiliza para garantizar el Peer es quién dice ser y no es suplantado. El ID debe renovarse periódicamente:

- El Peer puede usar un ID durante un minuto
- El Tracker aceptará un ID hasta dos minutos después de haberlo generado.

Por lo tanto, el Tracker debe almacenar en todo momento el ID signado a cada Peer; descartando los mensajes con un ID incorrecto.

La secuencia de mensajes Peer-Tracker es la siguiente:

1. El Peer envía al Tracker el mensaje "**connect request**".
    a. connection_id = 0x41727101980
    b. action = 0 // connect
    c. transaction_id = random
2. El Tracker recibe el mensaje "connect request" y valida lo siguiente:
    a. El tamaño mínimo del mensaje.
    b. connection_id = 0x41727101980
    c. action = 0 // connect
    d. Almacena transaction_id para la respuesta al Peer.
3. El Tracker responde al Peer con el mensaje "**connect response**".
    a. action = 0 // connect (si hay en error el Tracker envía 3 // error).
    b. trandaction_id = el que envió el Peer.
    c. connection_id = random (se almacena en el Tracker durante 2 minutos a modo de validación de la identidad del Peer)
4. El Peer recibe el mensaje "connect response" y valida:
    a. El tamaño mínimo del mensaje.

[1] Explicación basada en: http://www.bittorrent.org/beps/bep_0015.html y http://www.rasterbar.com/products/libtorrent/udp_tracker_protocol.html

b. trandaction_id = el que envió el Peer.

c. action = 0 // connect (el valor 3 representa un error).

d. Almacena el connection_id para intercambiarlo con el Tracker durante 1 minuto (pasado ese tiene que solicitar un nuevo connection_id mediante el intercambio de los mensajes "connect_resquest" y "connect_response".

5. El Peer envía al Tracker el mensaje "**announce request**" para solicitar una lista de Peers que forman parte de un determinado swarm. El Peer enviará este mensaje de forma periódica de acuerdo al intervalo devuelto por el Tracker en el mensaje "announce response". También se pude enviar el mensaje cuando se produzca un evento nuevo (completed, started, stoped) aunque no haya transcurrido el intervalo fijado por el Tracker. Además, cada vez que expire el connection_id el Peer debe obtener un nuevo ID antes de comunicarse con el Tracker.

   a. connection_id = el valor devuelto por el Tracker.

   b. action = 1 // announce

   c. transaction_id = el valor definido por el Peer en el "connect_request".

   d. info_hash = del contenido en torno al que se genera el swarm.

   e. peer_id = id del Peer.

   f. downloaded = número de bytes del contenido que ha descargado el Peer.

   g. left = número de bytes del contenido que le faltan al Peer.

   h. uploaded = número de bytes del contenido que el Peer ha compartido.

   i. event = 0: none; 1: completed; 2: started; 3: stopped. (Se inicia con "none").

   j. ip_address = 0 (poniendo 0 el Tracker usará la IP de origen del paquete UDP).

   k. key = random

   l. num_want = por defecto "-1". Alternativamente, el número máximo de Peers que se desea recibir.

   m. port = puerto en el que el Peer escuchará a otros Peers.

6. El Tracker recibe el mensaje "announce_request" y valida:

   a. El tamaño del mensaje.

   b. El connection_id y el transaction_id.

   c. action = announce.

   d. El tiempo transcurrido desde el último mensaje del Peer o el cambio del evento. Si no ha transcurrido el tiempo mínimo, el mensaje se descarta o se devuelve un mensaje de error.

7. El Tracker envía al Peer el mensaje "**announce response**" con una lista de Peers del swarm en el que está interesado el Peer.

   a. action = 1 // announce (si hay en error el Tracker envía 3 // error).

   b. transaction_id = El valor enviado originalmente por el Peer.

   c. interval = número de segundos que el Peer debe esperar antes de enviar un nuevo mensaje "announce request".

   d. leechers = número total de leechers del swarm.

   e. seeders = número total de seeders del swarm.

   f. El resto del contenido es una lista de tamaño variable con la siguiente estructura:

      - ip = codificado como un entero de 32 bits. Cada byte representa una parte de la dirección IP (0-255).

      - port = codificado como un entero sin signo de 16 bits.

8. El Peer recibe el mensaje "announce response" y valida:

   a. El tamaño del mensaje.

   b. El transaction_id.

   En este mensaje el Peer recupera la lista de pares IP + Puerto de los Peers con los que podrá comunicarse para intercambiar el contenido en que está interesado.

Estos 8 pasos definen la secuencia "normal" de mensajes intercambiados entre el Peer y el Tracker. Además, existen otros 3 mensajes:

- **error response**: Este es el mensaje que envía el Tracker al Peer si se produce algún error o alguno de los mensajes recibidos por el Tracker no son correctos.

   a. action = 3 // error

   b. transaction_id = El valor enviado originalmente por el Peer.

   c. message = cadena de texto con el mensaje de error.

- **scrape request:** mensaje que envía el Peer al Tracker para solicitar información de varios swarms: número de seeders, número de veces que se ha descargado el contenido y número de leechers. Se puede solicitar información de hasta 74 swams en un único mensaje "scrape request".

   a. connection_id = el valor devuelto por el Tracker.

b. action = 2 // scrape

c. transaction_id = el valor definido por el Peer en el “connect_request”.

d. El resto del contenido es una lista de tamaño variable con la siguiente estructura:

- info_hash-1
- ….
- info_hash-N

- **scrape response:** mediante este mensaje el Trackes envía al Peer la información de los swarms que se ha solicitado mediante un mensaje “scrape request”.

a. action = 2 // scrape (si hay en error el Tracker envía 3 // error).

b. transaction_id = el valor definido por el Peer en el “connect_request”.

c. El resto del contenido es una lista de tamaño variable con la siguiente estructura:

- complete = número de seeders del swarm
- downloaded = número de veces que se ha descargado el contenido.
- incomplete = número de leechers del swarm.

Notas para la implementación del Protocolo Peer-Tracker UDP

Para incorporar la funcionalidad del protocolo Peer-Tracker UDP el Tracker debe implementar una comunicación UDP que permita la recepción y envío de los diferentes tipos de mensajes descritos anteriormente.

En el proyecto “bitTorrentUtil” dentro del paquete “bitTorrent.tracker.protocol.udp” disponéis de un conjunto de clases que definen los diferentes tipos de mensajes. Dichos mensajes deben transformarse de objetos Java a bytes y viceversa, mediante los métodos **getBytes()** y **parse()**. Algunos de estos métodos están sin implementar; por ese motivo, el primer paso será dar contenidos a los métodos de los mensajes que no están implementados (tomando como referencia los que sí lo están, como por ejemplo los del mensaje ConnectRequest). Además, en el paquete “bitTorrent.tracker.protocol.udp.test” disponéis de un ejemplo de envío y recepción de los mensajes “connect request” y “connect response” a un Tracker público.

La comunicación Peer-Trackes se realizará mediante datagramas UDP con la siguiente particularidad:

- Todos los Trackers del clúster deben están unidos a un grupo de multicast (MulticastSocket), vinculado a la dirección IP que aparece en la URL “announce” del fichero “.torrent” procesado por el Peer.

- Los Peers envían los datagramas UDP a la dirección multicast. De esta forma, cada mensaje llega a todos los Trackers. El Peer no tiene que estar unido al grupo para poder enviar mensajes; simplemente usa la dirección IP de multicast como destino de los paquetes.

- Los mensajes que envían los Peers llegan a todos los Trackers (automáticamente por estar unidos a grupo de multicast), pero sólo el Máster responde.

- Con cada mensaje "announce request" el Peer solicita una nueva lista de Peers del swarm; y además notifica su interés en el swarm, el número de bytes descragados, restante y subidos. Está información se utiliza para actualizar la BBDD de cada Tracker; y en este punto se utiliza la funcionalidad de coordinación para almacenar la información en la BBDD. Del mismo modo, cuando expira un connection_id el tracker elimina al Peer correspondiente de la lista de Peers del swarm (y actualiza la BBDD).