**Implementación del Protocolo de Comunicación BitTorrent**

BitTorrent es un protocolo de intercambio de archivos peer-to-peer diseñado por Bram Cohen. Este protocolo está diseñado para facilitar la transferencia de archivos entre múltiples pares a través de redes poco confiables.

Este protocolo consta de 2 partes:

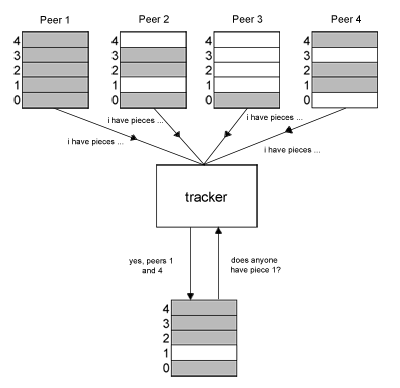
1. Se envía una solicitud al tracker y este responderá con una lista de pares o (peers). En otras palabras, de le dice al tracker el archivo que se desea descargar y este responde con la dirección ip de aquellos usuarios que ya han descargado el archivo en cuestión. Cada vez que se le hace un request al tracker, este agrega la dirección ip del solicitante a la lista, para que luego pueda ser utilizada en el protocolo y este nuevo solicitante pueda compartir el archivo una vez lo tenga descargado.
2. Una vez se tenga la lista de peers, se inicia la descarga a partir de la lista de los paquetes necesarios para completar el archivo.

Para el desarrollo de la implementación apostamos por **node.js,** ya que existe una gran cantidad de librerías con las cuales podemos llevar acabo la implementación de dicho protocolo.

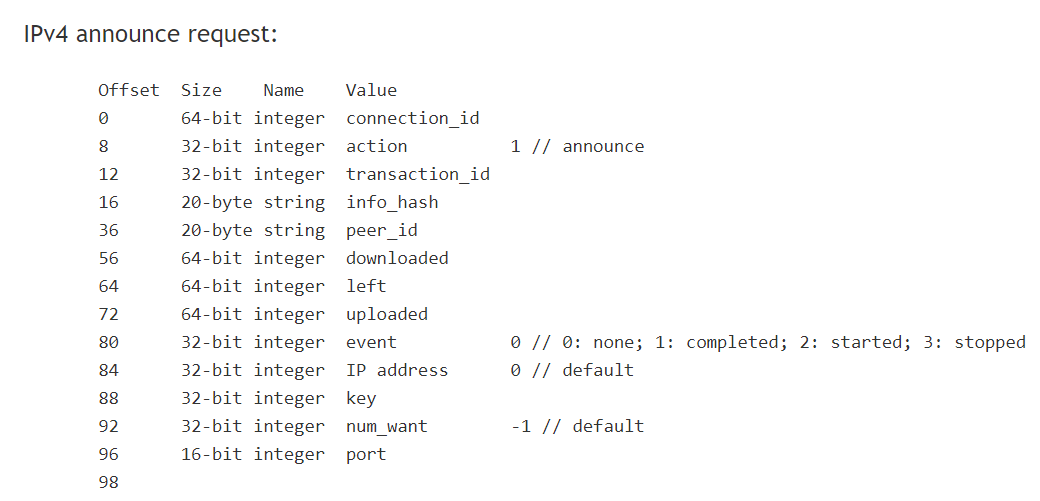
Para la codificación y decodificación del protocolo BitTorrent, utilizamos una librería la librería en **node** **Bencode**, ya que permite decodificar los datos trasmitidos poco estructurados.

Para el protocolo de comunicación en lugar de http, decidimos utilizar el protocolo **udp**, ya que este protocolo es el más utilizados tracker, por ofrecer un mayor rendimiento en la comunicación y enviado de los datos, sobre el protocolo http.

Para la implantación del manejo del protocolo udp, se utilizaron los módulos **dgram**, para el menejo de del protocolo, **url**, para extraer las informaciones de los tracker, como el puerto y hostname y el módulo **buffer**, para el manejo de los mensajes que se envía y se reciben atreves del dicho protocolo.



Cada mensaje obtenido a través del request, es decir cada vez que se lee el listado de peers con la descripción de las conexiones, se recibe como un búfer con un formato BitTorrent Enhancement Proposal (BEP). Utilizamos el método writeUInt32BE para ir leyendo cada elemento del buffer, acordar lo descrito en la documentación <http://www.bittorrent.org/beps/bep_0015.html>



Para el manejo del Hash, utilizamos el SHA1, ya que esta función de hash es la que usa bittorrent. Se utiliza el hash porque es una forma compacta de identificar de forma única el Torrent.

Para calcular el tamaño del archivo contenido en el torrent, se determinó que, si este solo contenía en la metada un archivo, se leería tamaño de dicho archivo como tal, con el cual iríamos comparando la cantidad de byte descargados. Pero en caso de ser un Torrent compuesto por varias partes, se leería la cantidad de archivos, la cual es un array de objetos, la cual luego se puede iterar por cada elemento y aplicar el mismo proceso. Como si este fuese un archivo.

Para la descarga del archivo creamos una conexión tcp con todos los pares de su lista. Luego de intercambiar algunos mensajes con el par como lo es la configuración, inicia a solicitar partes de los archivos que desee, los cuales son solicitadas y enviados por fragmentos. Estos fragmentos son almacenados temporalmente en memoria, para luego ser escribidos en el disco duro. Este proceso de mantiene dentro de una iteración, con cada elemento de la lista de peer hasta haber completado la descarga del archivo.

**Recursos:**

* [BitTorrentSpecification - TheoryOrg](https://wiki.theory.org/index.php/BitTorrentSpecification)
* [bep\_0015.rst\_post (bittorrent.org)](http://www.bittorrent.org/beps/bep_0015.html)
* [bep\_0020.rst\_post (bittorrent.org)](http://www.bittorrent.org/beps/bep_0020.html)
* [How to Write a Bittorrent Client, Part 1 | kristenwidman](http://www.kristenwidman.com/blog/33/how-to-write-a-bittorrent-client-part-1/)