BitTorrent

Xarxes Troncals i Serveis Públics de Dades

Marc Cosgaya Capel, u1959174

Universitat de Girona

1 Introducció

BitTorrent és el protocol més usat en pujada de fitxers[1]. Representa més d'un quart del trànsit de pujada en tot el món. No podem obviar la seva importància i, per tant, és interessant saber com funciona.

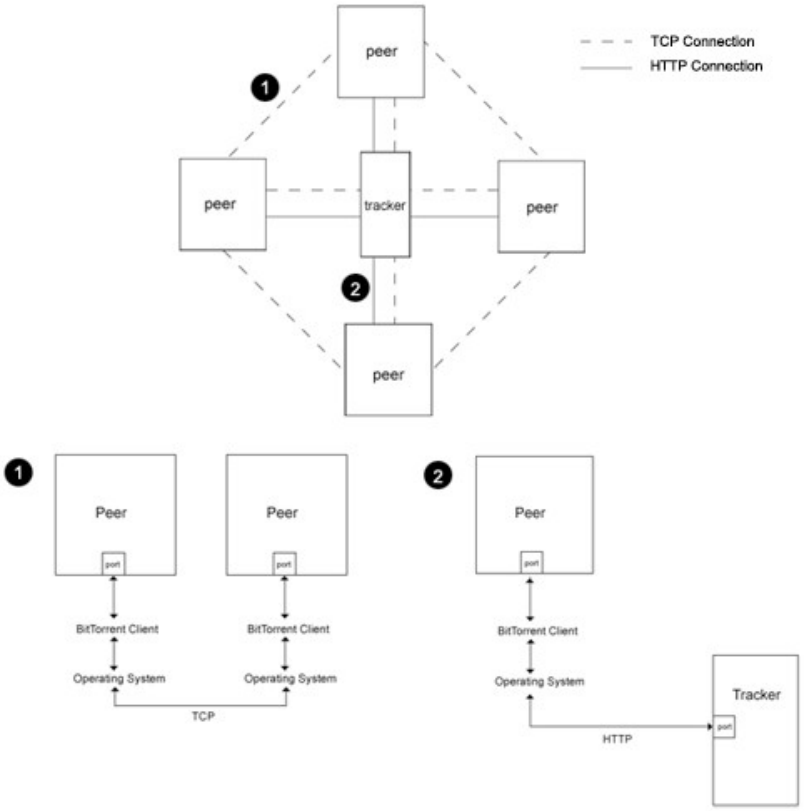
Aquest protocol va ser publicat el 2001 per Bram Cohen[2], un estudiant de la Universitat de Buffalo, EEUUA. Amb el temps s'ha convertit en uns dels protocols per compartir fitxers més prominent. La primera especificació es va establir el 2008[3], en el que es basarà l'explicació d'aquest text, i el segon el 2017[4], encara no estandaritzat.

La clau del protocol BitTorrent és distribuir la càrrega a l'hora de compartir els fitxers. Es divideixen els fitxers en fragments, on no tots provenen de la mateixa font, que després s'ajunten un cop s'ha acabat la descàrrega. Per fer això BitTorrent està estructurat en l'arquitectura de Peer to Peer (P2P) no-pur. No es tracta d'un P2P pur perquè necessita l'ús d'un servidor de senyalització.

2 Parts Implicades

Dins del protocol BitTorrent hi ha 3 elements o parts implicades. La primera és un **fitxer de metadades** amb extensió .torrent que conté la informació necessària per als futurs *peers*. La segona són els ***peers*** que transmeten i reben els fragments del fitxer, tot formant un eixam. La tercera és el servidor de senyalització o ***tracker*** que serveix per coordinar els *peers* dins d'una mateix eixam.

Primer, el futur *peer* obté el fitxer de metadades des d'un servidor web, per exemple. Després aquest fa servir les dades del fitxer per posar-se amb contacte amb el *tracker* corresponent. Aquest li respondrà la llista de peers a què s'hi pot connectar mitjançant el protocol BitTorrent en si per començar a fer les transmissions de fragments.



Interaccions entre el tracker i els peers. 1 és P2P i 2 és CS.[5]

2.1 Fitxer de Metadades

2.1.1 Codificació Bencode

Es fa servir una codificació anomenada Bencode. Aquesta es fa servir en els fitxers de metadades .torrent. És útil saber com funciona per poder interpretar-los.

- En aquesta codificació les *strings* estan codificades en forma de "mida:string". Per exemple, la codificació de la *string "*patata" és "6:patata".
- Els nombres enters estan codificats en forma de "iNe" amb N el nombre en base 10. Per exemple, el nombre 420 es codifica "i420e".
- Les llistes es representen amb "lXe" on X són *strings* i enters sense cap separador. Per exemple, la llista [2,"dies"] queda codificada "li2e4:diese".
- Els mapes o diccionaris queden codificats en forma de "dXe" on X és una alternació de claus i valors sense separadors. Una clau ha de ser una *string* i un valor pot ser una *string*, un enter o una llista. Per exemple, el diccionari {'dia':'dilluns','menjar':['patata','tomata']} s'escriu "d3:dia8:dilluns6:menjarl6:patata6:tomataee".

2.1.2 Contingut

El fitxer de metadades conté la informació necessària per començar. A dins hi ha un diccionari amb les claus "announce" i "info". "announce" la URL del *tracker*. "info" té altra informació rellevant, la més important és:

- "name": Té la suggerència de nom que fitxer o directori pot tenir en el sistema de fitxers local. No és un nom obligatori, es pot canviar.
- "piece length": Mida fixa dels fragments en què es divideix el fitxer original. Ha de ser una potència de 2.
- "pieces": Una *string* de mida múltiple de 20 que conté substrings de 20 bytes que es corresponen amb el hash SHA1 de cada fragment. El hash és útil per comprovar la integritat de cada fragment.
- "length": Mida del fitxer a descarregar en bytes.
- "files": Llista de diccionaris de fitxers. Cada diccionari té:
  - "length": Conté la mida del fitxer a descarregar en bytes.
  - "path": Llista dels directoris del fitxer. Per exemple "[directori,subdirectori,fitxer.png]".

Al fitxer de metadades hi pot haver "length" o "files" depenent de si s'està descarregant un únic fitxer o un conjunt de fitxers. També hi ha altres paràmetres, però els he obviat per simplificar l'explicació.

Hi ha *parsers* que permeten transformar el Bencode a altres formats més llegibles[6].

2.2 Tracker

El *tracker* és el servidor de senyalització encarregat de coordinar els *peers* dins d'un mateix eixam. Pot haver-hi més d'un a cada eixam. Aquest funciona amb HTTP i fa servir crides GET per obtenir i enviar la informació necessària. Les queries de la URL són les següents:

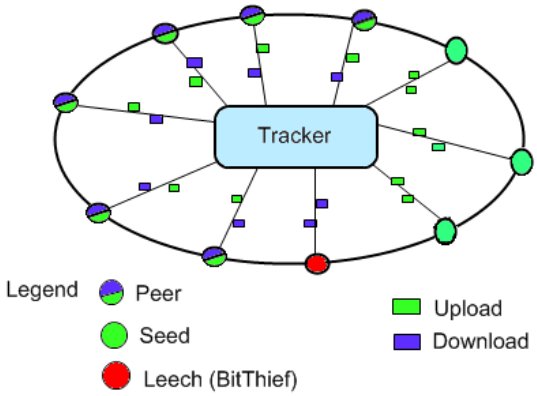
- "info_hash": Hash 1SHA de 20 bytes del fitxer de metadades. Serveix per verificar que s'està fent servir el fitxer correcte.
- "peer_id": Cadena de 20 caràcters a l'atzar que es genera a l'inici del protocol. Serveix per identificar al *peer* dins l'eixam.
- "ip": Ip del *peer*.
- "port": Port d'escolta del *peer* per rebre els paquets dels altres *peers*.
- "uploaded": Mida total carregada pel *peer*.
- "downloaded": Mida total descarregada pel *peer*.
- "left": Nombre de bytes que encara necessita per tenir el fitxer complet. Si és 0 es tracta d'un *seeder*, sinó es tracta d'un *leecher*.
- "event": String que indica l'estat del *peer*. Pot ser "started", "completed", "stopped" o "empty".

La resposta del *tracker* és en forma de *json* i conté:

- "interval": Temps regular d'espera, en segons, del *peer* entre peticions HTTP al *tracker*. El *peer*, però, pot enviar peticions no regulars sense respectar aquest interval.
- "peers": Llista de *peers* als quals es pot connectar i començar a rebre i transmetre els fragments. Cada *peer* té un identificador, una @IP associada i port.

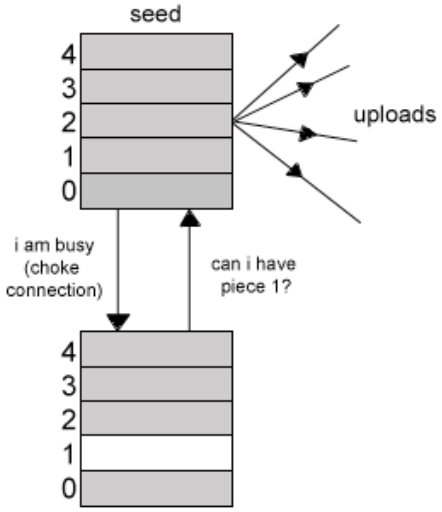
2.3 Peers

Els *peers* fan servir el protocol BitTorrent en si. Aquest funciona per sobre de TCP o uTP[7]. Dins l'eixam BitTorrent hi ha dos tipus de *peers* especials: sembradors, o ***seeders***, i xupadors, o ***leechers***[8]. Els *seeders* són els *peers* que ja han acabat la descàrrega i només es dediquen a compartir els fragments amb els altres. Els *leechers* són els peers que només descarreguen fragments sense compartir-ne cap, no són solidaris amb els altres. Els *seeders* no són els únics que envien fragments ja que els altres *peers* no-*leechers* també comparteixen els fragments que ja tenen.



Els tipus de peers segons la baixada o pujada de fragments.[9] L'eixam no té forma d'anella.

Els *peers* poden estar *choked* o no i poden estar *interested* o no. Quan un *peer* està *choked* vol dir que aquest ja té moltes connexions simultànies i demana als altres que no li demanin més fragments. Que estigui *interested* vol dir que informa als altres que vol començar a rebre fragments. Per tant, només hi ha transferència **si el receptor està *interested* i l'enviador no està *choking***.



Un peer interessat demana un fragment i l'altre informa que està choking.[10]

Al principi de la connexió hi ha el *handshaking* on els *peers* s'envien informació per determinar la legitimitat de la connexió que es vol establir. Comproven, per exemple, que el *hash* SHA1 rebut correspongui amb el del fitxer de metadades i que "peer_id" correspongui amb la @IP correponent que s'ha rebut anteriorment del *tracker*. Després comença el diàleg de la transferència de fragments.

A vegades, els *peers* s'envien paquets amb *payload* buida per mantenir la connexió activa (keep alive). També, quan un *peer* obté un fragment compara el *hash* SHA1 generat amb el corresponent de la llista del fitxer de metadades. Si no coincideixen, el descarta.

Els paquets del protocol són:

- Paquet Handshake:

Prot. Name Length (1)	Protocol Name (19)	Reserved Bytes (8)	Metainfo SHA1(20)	Peer ID (20)

- Paquet Choke:

Message Length = 1 (4)	Message Type = 0 (1)

- Paquet Unchoke:

Message Length = 1 (4)	Message Type = 1 (1)

- Paquet Interested:

Message Length = 1 (4)	Message Type = 2 (1)

- Paquet Not Interested:

Message Length = 1 (4)	Message Type = 3 (1)

- Paquet Request:

Message Length = 13 (4)	Message Type = 6 (1)	Piece Index (4)	Begin Offset (4)	Length (4)

- Paquet Piece:

Message Length (4)	Message Type = 7 (1)	Piece Index (4)	Begin Offset (4)	Data (N)

Hi ha altres tipus, com *have* que indica que s'ha rebut correctament o *bitfield* que indica quins fragments ja té.

2.3.2 NAT Traversal

Partint de la pregunta que se m'ha fet a classe sobre la problemàtica dels peers sota NAT, voldria comentar que existeix una solució[11]. En aquesta solució es fa servir un *relaying peer* que no està sota NAT i que coordina la connexió de dos altres *peers* sota NAT. Això s'aconsegueix fent servir missatges *rendezvous*. Tanmateix, si no hi ha cap *peer* sense NAT és impossible.

3 Pros i Cons[12]

Avantatges:

- La descàrrega no depèn d'una única font. Si una connexió entre *peers* cau, el servei no queda interromput.
- La velocitat és més ràpida. No està limitada peel *bitrate* del servidor en C-S. Normalment, un servidor HTTP no només està servint una única descàrrega o petició.
- Integritat dels fitxers. És difícil descarregar un fitxer alterat perquè es fan servir *hash* per comprovar la integritat dels fragments.

Desavantatges:

- Depèn del nombre de *seeders*. Si no n'hi ha, és impossible iniciar la descàrrega. De fet, la velocitat de la descàrrega depèn de la quantitat de *seeders*. Es diu que un eixam és "saludable" quan hi ha molts *seeders*[13].
- Trànsit elevat. El protocol fa servir molt del *bitrate* i pot ser que peticions com carregar una pàgina web triguin molt a ser servides.
- Exposa @IP pública. Es necessita una @IP pública per establir la connexió i pot ser font de problemes amb la seguretat. Això pot ser evitat mitjançant un túnel VPN.

Bibliografia

1: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf

2: https://web.archive.org/web/20080129085545/http://finance.groups.yahoo.com/group/decentralization/message/3160

3: https://www.bittorrent.org/beps/bep_0003.html

4: https://www.bittorrent.org/beps/bep_0052.html

5: https://www.morehawes.co.uk/the-bittorrent-protocol

6: https://www.npmjs.com/package/parse-torrent-file

7: https://www.bittorrent.org/beps/bep_0029.html

8: https://ccm.net/faq/29775-what-are-seeders-and-leechers

9: https://blogs.umass.edu/Techbytes/2015/04/28/bittorrent-an-explanation-of-the-protocol/

10: https://www.morehawes.co.uk/the-bittorrent-protocol

11: http://bittorrent.org/beps/bep_0055.html

12: https://ergonotes.com/pros-and-cons-of-using-torrent-clients

13: http://wiki.vuze.com/w/Torrent_health