Úvod

BitTorrent (dále jen BT) je protokol pro distribuci souborů, založený na peer-to-peer P2P, což znamená, že člověk který stahuje určitý soubor (peer, v terminologii BT leech) ho poskytuje/uploaduje dalším lidem (rozuměj části, co už stáhnul). Tento protokol je vhodný pro distribuci souborů mnoha lidem v krátkém okamžiku. A je mnohem levnější a efektivnější, než použití tradičních centralizovaných protokolů (HTTP/FTP/...). Jeho vyjímečností je dobře vyřešená a hlavně fungující FairPlay pravidla pro zaručení, že lidé co dostatečně uploadují dalším peerům, stáhnou soubor velmi rychle. Také se dá zjednodušeně říct, že rychlost uploadu přímo úměrně ovlivňuje rychlost downloadu. Jeho nevýhodou je vymírání starších torrentů. O určitý soubor postupně opadává zájem a klesá počet lidí sdílející ho (tito lidé tvoří v term. BT tzv. Swarm), a může se stát, že v celém swarmu neexistuje kompletní kopiie daného souborů, a to ani pokud se složí všichni účastníci(což je plně postačující). Peeři, kteří stáhli soubor, a přesto ho ještě sdílí, se nazývají seedeři, a tito jsou zárukou, že pokud jsou připojení, torrent nevymře, protože oni vlastní kompletní kopii.

Motivační příklad komečního využití

Nejrozšířenější MMORPG hra současnosti se jmenuje World of Warcraft od společnosti Blizzard Entertainment. V těchto dnech ji hraje 10 mil. platících zákazníku. Aby tato hra udržela zájem hráču, poměrně pravidelně vydává updaty (říká se jim patche), které přidávají nový obsah(nové lokace, zbroj, ...) a vybalancovávají hru. Větší patche přícházejí každé 2 měsíce a mají velikost 60MB – 1GB. Po vydání patche se dá do hry přistupovat jen s danou nejnovější verzí hry, a proto je nutné, aby daný patch měli zákazníci do několika hodin stažený a nainstalovaný, a to ze dvou důvodů, protože se platí měsíční poplatky, a také je není možné nutit, aby měli počítač zapnutý déle než chtějí (aby stáhli patch).

Vezměme příklad patch velikosti 1GB pro 3.5 mil. evropských pařanů, což je nutnost přenést 3.5 EB(E je Exa a je to 1000TB, ale také 1 000 000 000 000 000 000B) dat během řekněme 5ti hodin.

Pokud by zvolili centralizovaný způsob distribuce, bylo by potřeba přípojky o rychlosti 1.5TBps nebo také 1500 serverů s 1Gb-tovou síťovkou. Největším problémem, ale není hardware, ale poplatky za připojení a trafic. Poskytovatelem Blizzardu v Evropě je TeliaSonera, jejíž páteřní spoje jsou nejvíce OC192, což je přibližně 10Gbit (stahování by trvalo více než měsíc). Proto se patche distribuují přes torrent a vlastního klienta (program implementujicí protokol, tedy ten program, kteří peeři používají). K největším patchům je poskytován i .torrent file (viz. dále).

Postup uživatele používajícího BitTorrent

Uživatel využije nějakou vyhledávací službu na torrenty (nepříklad minova, isohunt, či dnes už zakázaný demonoid.org). Z té stáhné .torrent soubor, který obsahuje metadata o souboru(ech), který/é chce stahovat, například jméno, velikost, popis, jména trackerů (viz déle). Tento soubor otevře ve svém oblíbeném klientů a z trackerů stáhne seznam peerů(lidí stahující/sdílející dané soubory) v daném swarmu. Klient se k peerům postupně připojuje a žádá je o poskytnutí částí. Výběr částí a to jestli je soubor poskytnut se řídí speciálními algoritmy, které zvyšují efektivitu a udržují FairPlay (viz dále).

Shrnutí názvosloví

- **peer** účastník sdílení/distribuce
- **leech** účastník, který nevlastní plnou kopii souboru (také někdy označení pro neslušné účastníky)

- **seed** účastník vlastnící plnou kopii souboru, který tedy jen uploaduje
- **.torrent** soubor popisující soubor/y, které chci stáhnout
- **swarm** seznam peerů učastnících se na jednom torrentu
- **tracker** vyčleněný a centralizovaný server, schraňující a distribující seznam peerů
- **klient** program, který implementuje torrent protokol
- **announce** klientovo první oznamení trackeru, že se chce přidat do seznamu peeru v daném swamu
- **scrape** pravidelný dotaz klienta na tracker pro aktualizování statistik swarmu a seznamu peerů

Historie

Protokol uvedl v roce 2001 Bram Cohen, který založil společnost BitTorrent, Inc., starší verze protokolu a referenčního klienta jsou open source, ale novější budou uzavřené.

Struktury a formáty

Bencoding

V protokolu je využíván tzv bencoding, který zajišťuje jednoduchý a crossplatform přenos řetězců, celých čísel, seznamů a slovníků. Jednotlivé prvky jsou samooddělující.

- **string** <délka jako ASCII>**:**<text řetězce>
- **integer** “i“<ASCII text čísla>“e“
- **list** “l“<prvky řazené za sebou>“e“
  - například: *l4:spam4:eggse* odpovídá *["spam", "eggs"]*
- **dictionary** “d“<bencoded string><bencoded element>“e“
  - například: *d3:cow3:moo4:spam4:eggse* odpovídá *{ "cow" => "moo", "spam" => "eggs" }*

.torrent file

Je zakódován pomocí bencoding a stringy jsou kódováný v UTF-8.

Na nejvyšší úrovni je tvořen pomocí dictionary:

- **info** dictionary popisující soubor(y) v torrentu, mírně se liší podle toho, zda je torrent pro jeden či více souborů
- **announce** URL trackeru
- **announce-list** nepovinný seznam trackerů (nahrazuje předchozí announce)
- **creation date** nepovinný čas vytvoření v „UNIX epoch format“ (integer od 1. ledna 1970 UTC)
- **comment** nepovinný popis
- **created by** nepovinné jméno a verze programu, který vytvořil tento torrent soubor

Info dictionary

- **piece length** velikost bloků určena jako optimální pro velikost stahoveného souboru a

torrent souboru

- **pieces** jeden string složený z 20bytových SHA1 hashů jednotlivých bloků, více souborů se považuje za kontinuální stream
- **private** nepovinný integer popisující: **1**, že peer musí svojí existenci zveřejňovat jen a pouze trackerům uvedených v tomto torrentu. (nesmí používat DHT,PEX, atd...)

Single file Info dictionary

- **name** jméno souboru
- **length** integer délky souboru
- **md5sum** nepovinný MD5 hash souboru jako 32 znakový hexadecimální string

Multiple file Info dictionary

- **name** jméno adresáře obsahujícího všechny soubory
- **files** seznam(list) dictianary pro jednotlivé soubory
    - **length** délka souboru
    - **md5sum** nepovinný MD5 hash souboru
    - **path** jméno a cesta k souboru kódovaná jako list jednotlivých elementu cesty (jeden string na každý adresář plus jméno souboru)

Komunikace tracker<->peer

Tracker je služba pracující na HTTP/HTTPS protkolu (více v RFC1738).

Požadavky se zasílají pomocí GET, parametry požadavku jsou umístěné v URL za znakem otazníku(před je adresa trackeru z announce části .torrent souboru) ve formátu param=value a jednotlivé parametry se oddělují přes ampersand, value je nutné kódovat, aby mohlo být umístěno v URL (viz dané RFC, metoda se označuje urlencode).

- **info_hash** 20 bytový SHA1 hash *info* části .torrent souboru
- **peer_id** 20 znakové string identifikující klienta
- **port** naslouchací port
- **uploaded** počet uploadovaných bajtu od eventu started jako string čísla v desítkové soustavě
- **downloaded** počet stáhnutých bajtu
- **left** počet bajtů, které klient chce ještě stáhnout
- **compact** indikuje, že klient dokáže přijímat compact response
- **no_peer_id** indikuje, že tracker může vypustit peer_id v seznamu peerů, nastaveni je ignorováno, pokud je použit compact response
- **event** pokud je použit musí být jeden z následujících (nebo prázdný):
    - **started** první připojení k trackeru v rámci daného torrentu
    - **stopped** zasílá se trackeru pokud je klient ukončen správně
    - **completed** zasílá se pokud je stažen kompletní soubor

  pokud není parametr uvedený, nebo je prázdný, považuje se zpráva jako pravidelný dotaz na nové peery
- **ip** nepovinná IP adresa, která je například nutná pokud klient komunikuje s trackerem přes

proxy, nebo tracker i klient jsou na stejné straně NAT gateway(a externí peeři před NAT ji musí využit). Parametr je string: pro IPv4 tečkama oddělený quad(byte), pro IPv6 hexadecimalní zápis podle rfc3513.

- **numwant** nepovinný počet peerů, který klient chce, může být nula, pokud vynechán bere se většinou jako 50.
- **key** nepovinná další možnost zabezpečení, která není sdílená s ostatními uživateli, umožnuje také změnu IP při zachování identity.
- **trackerid** nepovinný, pokud byl získán z předchozí odpovědi, měl by být v dalších vyplněný

Odpověď, tracker odpoví zasláním dokumentu(tak jako normální web server přes HTTP) s „text/plain“ MIME, který je zakódován v bencode. Parametry (bencoded dictionary):

- **failure reason** v případě chyby, ostatní parametry by měly být vynechány, měl by obsahovat lidsky čitelný důvod chyby
- **warning message** nepovinny lidsky čitelný text nezávažné chyby, která nebrání pokračování
- **interval** doporučený interval v sekundách, kdy by měl klient zaslat znova dotaz na peery
- **min interval** nepovinný, minimální interval v sekundách, kdy klient může znovu žádat o scrape(nové peery)
- **tracker id** string, který posíla klient zpět trackeru v dalších požadvcích, zůstává perzístentní i pokud není znovu uveden
- **complete** počet peerů vlastnících celou kopii (seederů)
- **incomplete** počet peerů nevlastnící celou kopii
- **peers** *(slovníkový/dictionary model)* list dectioneries:
    - **peer id** ID daného peera, které si sám vybral, a které zaslal v announce
    - **ip** IPv4 tečková notace, IPv6 hexadecimální notace nebo DNS jméno
    - **port** integer naslouchací port daného peera
- **peers** *(binární model)*, používa se při compact nastavení, zřetězení 6bajtových stringu obsahujících 4 bajty na adresu a 2 bajty na port, vše v síťové(big endian) notaci

Peeři jsou vybíráni většinou náhodně, ale některé implementace mají inteligentnější chování, například nezasílají seznam seederů seederům

Scrape request, je další typ požadavku, který většina trackerů konvencí podporuje. Umožňuje lehce získat statistiky daného torrentu, a umožňuje vyhledávacím službám lehce aktualizovat informace. URL se vytvoří nahrazením slova announce slovem scrape. Jediný parametr je info_hash, kterých může být více, a kterým se vybírá určitý torrent (viz normální announce request), pokud není uveden jsou vráceny všechny torrenty, které tracker spravuje.

Odpověď' je podobná normální odpovědi, text/plain bencoded soubor, který je někdy zagzipovaný. Obsahuje:

- **files** slovník souborů indexovaných pomocí hash_info: 20bajtový SHA1 hash info hlavičky torrent souboru, hodnotou je slovník:
    - **complete** počet seederů
    - **downloaded** celkový počet registrovaných complete eventů
    - **incomplete** počet leechu, peeru s nekompletní kopií

- **name** nepovinné, interní jméno uvedené v torrent souboru

Komunikace peer<->peer

Je prováděn přes TCP. Dále bude rozlišován piece/kousek, který je uveden v torrent souboru, a block, což je jednotka pro výměnu.

Na každe otevřené spojení s jiným peerem je nutné udržovat informace:

- **choked** s daným peerem nechci komunikovat, a jeho požadavky budu ignorovat, a proto klient na druhé straně by se o ně neměl ani snažit
- **interested** indikuje, že se vzdáleným peerem chci komunikovat, protože má části, které chci, indikuje, že s nim budu komunikovat až nebudu choked

Klient teda musí ukládat 4 informace pro každého vzdáleného peera, protože komunikace je obousměrná (am_choking, am_interested, peer_choking, peer_interested). Komunikace začíná se stavy (am_choking=1, am_interested=0, peer_choking=1, peer_interested=0). Blok se může stahovat, když jsem interested k danemů vzdálenému peeru a on je ve vztahu ke mně unchoked. Blok se může uploadovat, když je vzdalený peer interested a já ho mam unchocked.

Formáty zpráv: neníli řečeno jinak integer je 4bajtový kódovaný pomocí bigendian. Všechny zprávy jsou zasílany s délkovým prefixem označujícím délku následující zprávy. Komunikace peer-peer začíná zprávou **handshake**, která má 49+len(pstr) délkový prefix:

- **pstrlen** délka následujícího řetězce (19)
- **pstr** textová identifikace protokolu **„**BitTorrent protocol “
- **reserved** 8 bajtu k budoucímu použití, aktualní implementace je všechny nuluje, budou použity ke změně chování protokolu a nastavování nových funkcí, které nebyli v původním protokolu
- **info_hash** 20 bajt SHA1 hash info části torrentu, který je požadován
- **peer_id** 20 bajt jedinečné identifikace peera, mělo by být stejné jako, které bylo zaslané trackeru, azareus posílá jiné v případě anonymního nastavení.

Pokud dorazi info_hash, který klient neobsluhuje je spojení zahozeno, pokud dorazí neočekávane peerid(získané z trackeru) spojení je iniciátorem zahozeno.

Peer_id: existují dvě konvence na vytváření peerid, ve kterém je zakódováno jméno a verze klienta: Azureus-style a Shadow's-style.

Azureus-style: '-', dvě písmena pro jméno klienta, 4 ASCII čísla pro jeho verzi, '-', a dále náhodná čísla.

Někteří (nejznámější) klienti:

- **AZ** Azureus
- **BC** BitComet
- **SZ** Shareaza
- **UT** µTorrent

Shadow's-style: jeden alfanumericky ASCII znak pro jméno klienta, 5 znaků pro verzi zarovnaný pomocí '-' pokud je kratší než 5 znaků, následovaný třemi znaky '-', následovaný náhodnými znaky. Číslo verze se označuje písmeny pokud je vetší než 9: 'A' = 10, 'Z' = 35, 'a'=36, ... Například '*S58B*-----' je Shadow 5.8.11 .

Poté existuje mnoho klientů, kteří používají vlastní konvenci.

Další zprávy: drží se formátu: <length prefix><message ID><payload>, první je 4 bajtový bigendian integer, druhý je jednoznakové decimální číslo:

- **keep-alive: <len=0000>** jako jediná má délku nula a žádný id ani payload, pokud klient nedostane po určitý čas žádná data má možnost zavřít spojení. Keep-alive se posílá každé dvě minuty, pokud nebyly zaslány, během dané doby, žádné jiné zprávy.
- **choke: <len=0001><id=0>**
- **unchoke: <len=0001><id=1>**
- **interested: <len=0001><id=2>**
- **not interested: <len=0001><id=3>**
- **have: <len=0005><id=4><piece index>** obsahuje index od nuly aktuálně staženého kousku/piece (který měl správně hash)
- **bitfield: <len=0001+X><id=5><bitfield>** zasílá se na začátku spojení a obsahuje bitovou mapu vlastněných kousků se správným hashem, přebývající bity jsou vyplněné nulou
- **request: <len=0013><id=6><index><begin><length>** požadavek na download dat
    - **index** kousku(pices) od nuly, který chci stáhnout
    - **begin** offset v rámci kousku
    - **length** délka kolik chci stáhnou, implementace uvádějí, že pří překročení určité velikosti se má ukončit spojení, tato velikost se mění a není ustálená, doporučuje se 2^14 – 16KB, předpokládá se tedy, že bloky jsou menší než piecies
- **piece: <len=0009+X><id=7><index><begin><block>** obsahuje data daného bloku, je to tedy odpověď na předchozí zprávu, X představuje délku bloku
- **cancel: <len=0013><id<=8><index><begin><length>** zruší request a používá se při End Game(viz. Dále)
- **port: <len=0003><id=9><listen-port>** informuje o naslouchacím UDP portu pro DHT

Algoritmy

Super-seeding používá se k nastartování daného torrentu, když existuje pouze jeden seeder(ten co vytvořil daný torrent) a ostatní nemají nic. Daný seeder se pak tváří jako normální peer, který nemá nic, a po vytvoření spojení s jiným peerem pošle zprávu have s indexem kousku, který ještě nikomu neuploadoval, peer na druhé straně je tedy nucen stahovat tento kousek, další kousky jsou mu zaslany jen když jiný(třetí) peer oznamí, že tento kousek má, což znamena že vzdálený peer dané kousky distribuuje dál, a je tedy efektivní mu zasílat další. Při normálním seedovaní musí iniciální seeder uploadovat min. 150-200% velikosti souboru, aby se jiný peer stal také seederem. Při super seeding stačí odeslat 105% velikosti. Tento režim by se měl používat jen, pokud existuje jen prvotní seeder, jinak omezuje efektivitu protokolu, protože peeři nemužou stahovat kousky, které vlastní jen z části (tudiž nemají platný hash kousku a nemouhou ho uploadovat jiným).

Strategie výběru kousku: v nejjednoduší variantě mohou klienti vybírat kousky v **náhodném pořadí,** ale pro protokol je efektivnější, pokud jsou vybírány **rarest-first.** Pro každého peera je tedy udržována aktuální mapa vlastněných kousku ze zpráv have a bitfield. Dále je efektivní, aby bylo vybráno z několika nejvzácnějších kousků náhodně, aby se zamezilo, že všichni budou chtít stejný nejvzácnější kousek.

End-Game: ke konci stahování mohou poslední bloky docházet pomalu, proto klient pošle požadavky na dané bloky více klientům, a pokud ho stáhne, všem ostaním zašle zprávu cancel.

Choking a Optimistic Unchoking: přiškrcování se provádí, protože TCP „congestion control“ funguje špatně při řízení více spojení najednou, a aby mohly být implementováný reciproční algoritmy, a hlavně pro zajištění dobrého downloadu.

Aktuální používaný algoritmus funguje následovně: 4 peeři, kteří mají nejlepší upload a jsou interested vůči mě jsou unchoked, tím se zaručí moje vysoká rychlost downloadu, tito jsou pak označeni jako downloaders.

Peeři, kteří mají lepší upload rate vůči mě, ale nejsou interested jsou také unchocked. Pokud se stanou interested nejhorší peer z downloaders je nahrazen. Pokud se stanu seederem, tak meřítkem přestává být upload ke mně ale ode mě.

V **optimistic unchoking** je vybrán ještě jeden peer bez ohledu na upload ke mně (nebo ode mě při seedování), pokud je interested přibude jako páty k downloaders, tito peeři rotují každých 30 sekund. Nově připojení peeři dostávají 3x větší šanci na optimistic unchoking. Tento algoritmus se stará o objevování nových nejlepších peerů.

Nejvíce implementovaná rozšíření

Distributed Hash Table (DHT): využití této extension se signalizuje v handshake v reserved[7] nastavené jako |= 0x01, klient/peer implementující DHT se stává node. Pote není vůbec potřeba tracker, prvotní množina nodů je zapsána v .torrent souboru v části nodes, v terminologii bencode je to list dvojic(vnořený list): IP/DNS jméno a UDP naslouchací port. Algoritmem a implementací je Kademila. V DHT jsou uloženy všechny torrenty a jsou indexované pomocí SHA1 hashe info části. Metrikou pro vyhledávání je info_hash XOR peer_id.

Peer exchange (PEX): je vytvořeno malé DHT jen pro daný torrent(swarm). Pokud nějaký klient chce přidat nového peera, vygeneruje náhodné číslo, a hledá v daném DHT peera s nejbližším id.

Fast Peers Extensions: je možno brát jako soubor extensions, přidává mnoho nových zpráv:

- **Have None: <len=0x0001><op=0x0E>** peer nemá žádné kousíčky
- **Have All: <len=0x0001><op=0x0F>** peer je seeder, oboje dvoje se objevují hned po handshake a nahrazují bitfield
- **Reject Requests: <len=0x000D><op=0x10><index><begin><offset>** informuje vzdálený peer, že jeho request nebude obsloužen
- **Suggestions: <len=0x0005><op=0x0D><index>** měl bys stahovat tento kousek, používá se pro super-seeding a pro omezení seeku na disku, může jich být zasláno více, a pak si klient může vybrat. Seeděři implementující protokol musí dbát, aby sít dostala všechny kousky rovnoměrně.
- **Allowed Fast: <len=0x0005><op=0x11><index>** je pro nově připojené peery do swarmu, aby nebyly biti tit-for-tat(recipročními) algoritmy, a znamená dám ti tento kousek, i když si chocked, ale nemusí znamenat, že je daný peer vlastní, čísla kousků(setu kousků) jsou generovány standardizovaným algoritmem, který vygeneruje stejný set(velikosti *k*) od každého vzdáleného peera, tímto je zabráněno zneužití. Vzdálený peer by měl zaslat kousek, který ještě peer nemá a je obvykle, že reqeust je odmítnut, pokud peer vlastní více než *k* jakýchkoliv kousků.

Kryptování spojení (PHE): používá se hlavně pro maskování torrentového trafiku mezi peery vůči zlým ISP. Používa RC4 kryptování a přidává náhodná data pro zamezení detekce na základě analýzy délky zpráv. Je možné kryptovat jen hlavičky či celý tok.

Implementace

- **hardware** některé routery mají klienta integrovaného v sobě
- **BitTorrent** oficiální implementace autora protokolu, napsaná v Pythonu a existuje pro Windows, Mac OS X i Linux
- **µTorrent** velmi nenáročný na zdroje a kvalitní klient napsaný v C++ pro Windows o velikosti 200KB
- **Azureus** je opensource projekt napsaný v jave, proto je přenositelný do všech prostředí, kde běží java
- **Opera** webový prohlížeč, který od verze 9 má v sobě implementovaného klienta

Nevýhody

Je poměrně snadno detekovatelný i přes použití kryptování. Dochází k vymírání (viz výše). Může otevírat velmi velké množství TCP spojení, stává se to, ale jen při neohleduplném nastavení klienta, protože defaultní implementace říka, že tracker má zaslat jen seznam maximálně 30ceti peeru, a stahování pak probíha jen od 4 downloaders + 1 v případě optimistick choocking + několik seederu. Jediné co lze vytknout, že s peerama, se kterýma nekomunikuju, musím být připojen otevřeným TCP kanálem. Další nevýhodou je špatná decentralizace, co se týče vyhledávací služby, DHT a PEX slouží jen k vyhledávání nových peerů. S tímto problémem se ale potýkají většinou jen trackery se zaměřením na ilegalní obsah.

Použité zdroje:

Doporučuji (s využitím odkazů v tomto článku pro daná rozšíření)

http://wiki.theory.org/BitTorrentSpecification

Dále

http://en.wikipedia.org/wiki/BitTorrent_%28protocol%29

Vytvořil: Martin Fúsek, 1.1.2008