Hálózatba kapcsolt erőforrás platformok és alkalmazásaik

Simon Csaba

TMIT

2017

BitTorrent

BitTorrent

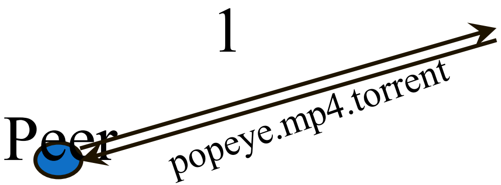
- Written by Bram Cohen (in Python) in 2001
- "Pull-based" "swarming" approach
  - Each file split into smaller pieces
  - Nodes request desired pieces from neighbors
    - As opposed to parents pushing data that they receive
  - Pieces not downloaded in sequential order
- Encourages contribution by all nodes

What is BitTorrent?

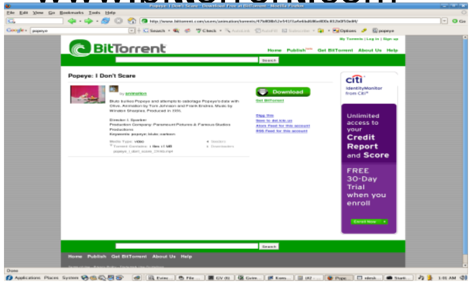
◆ Efficient content distribution system using *file swarming*. Does not perform all the functions of a typical p2p system, like *searching*.

◆ The throughput increases with the number of down loaders via the efficient use of network bandwidth

How does a node starts the download?



www.bittorrent.com


Popeye: I Don't Scare - Download Free at BitTorrent - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
BitTorrent
My Torrents | Log in | Sign up
Home Publish Get BitTorrent About Us Help
Search
Download
Get BitTorrent
Digg this
Atom Feed for this account
RSS Feed for this account
citi
IdentityMonitor from Citi
Unlimited access to your Credit Report and Score
FREE 30-Day Trial when you enroll
Enroll Now
Done
Applications Places System

Popeye: I Don't Scare

by animation

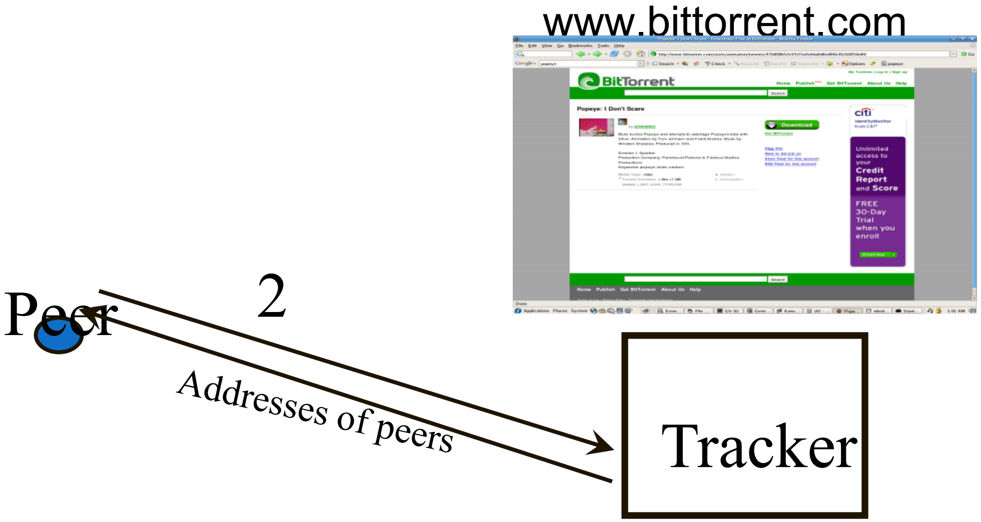
Bluto bullies Popeye and attempts to sabotage Popeye's date with Olive. Animation by Tom Johnson and Frank Endres. Music by Winston Sharples. Produced in 1935.

Director: I. Sparber
Production Company: Paramount Pictures & Famous Studios Productions
Keywords: popeye; bluto; cartoon

Media Type: video
Torrent Contains: 1 files 11 MB
popeye_i_dont_scare_1934b.mp4

- An „**index file**” *popeye.mp4.torrent* hosted at a **(well-known) webserver**

How does a node starts the download?



www.bittorrent.com

Popeye: I Don't Scare

by animation

Bluto bullies Popeye and attempts to sabotage Popeye's date with Olive. Animation by Tom Johnson and Frank Endres. Music by Winston Sharples. Produced in 1935.

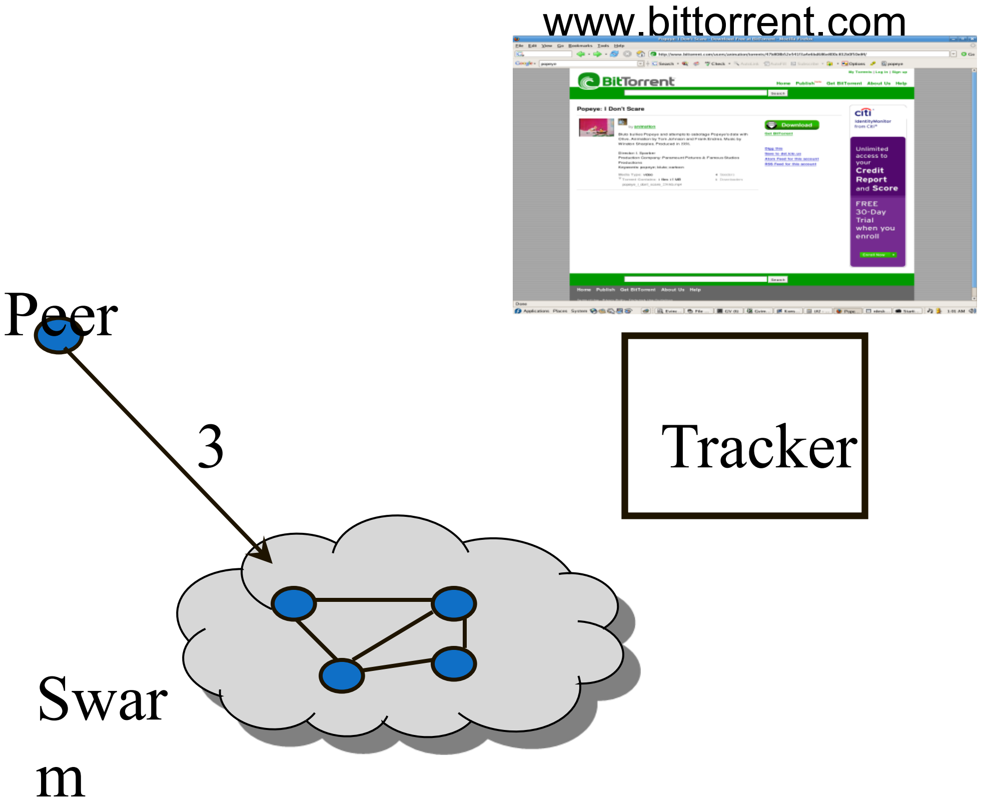
Director: I. Sparber

Production Company: Paramount Pictures & Famous Studios Productions

Keywords: popeye; bluto; cartoon

- An „**index file**” *popeye.mp4.torrent* hosted at a **(well-known) webserver**
- The .torrent has address of **tracker** for file

How does a node starts the download?


www.bittorrent.com
BitTorrent
Home Publish Get BitTorrent About Us Help
Popeye: I Don't Scare
Download
citi
Unlimited access to your Credit Report and Score
FREE 30-Day Trial when you enroll
Peer
3
Tracker
Swar
m

- An „**index file**” *popeye.mp4.torrent* hosted at a **(well-known) webserver**
- The .torrent has address of **tracker** for file
- The tracker, which runs on a webserver as well, keeps track of **all peers downloading** file

File sharing

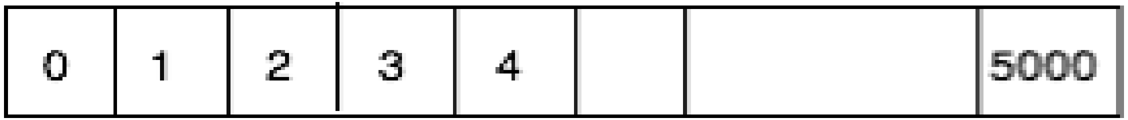
To share a file or group of files, the initiator first creates a **.torrent** file, a small file that contains

- **Metadata** about the files to be shared
- SHA-1 hashes of each piece in file - for reliability
- “files” – allows download of multiple files
- Information (e.g., URL) about the **tracker**, the computer that coordinates the file distribution.

Downloaders first obtain a **.torrent** file, and then connect to the specified **tracker,** which tells them from which other peers to download the pieces of the file.

How it works

The file to be distributed is split up into pieces and an SHA-1 hash is calculated for each piece



BT Components

The peers first obtain a metadata file for each object
The metadata contains:

- The SHA-1 hashes of all pieces
- A mapping of the pieces to files
- Piece size
- Length of the file
- A tracker reference

BT Components

The **tracker** is a central server keeping a list of all peers participating in the swarm

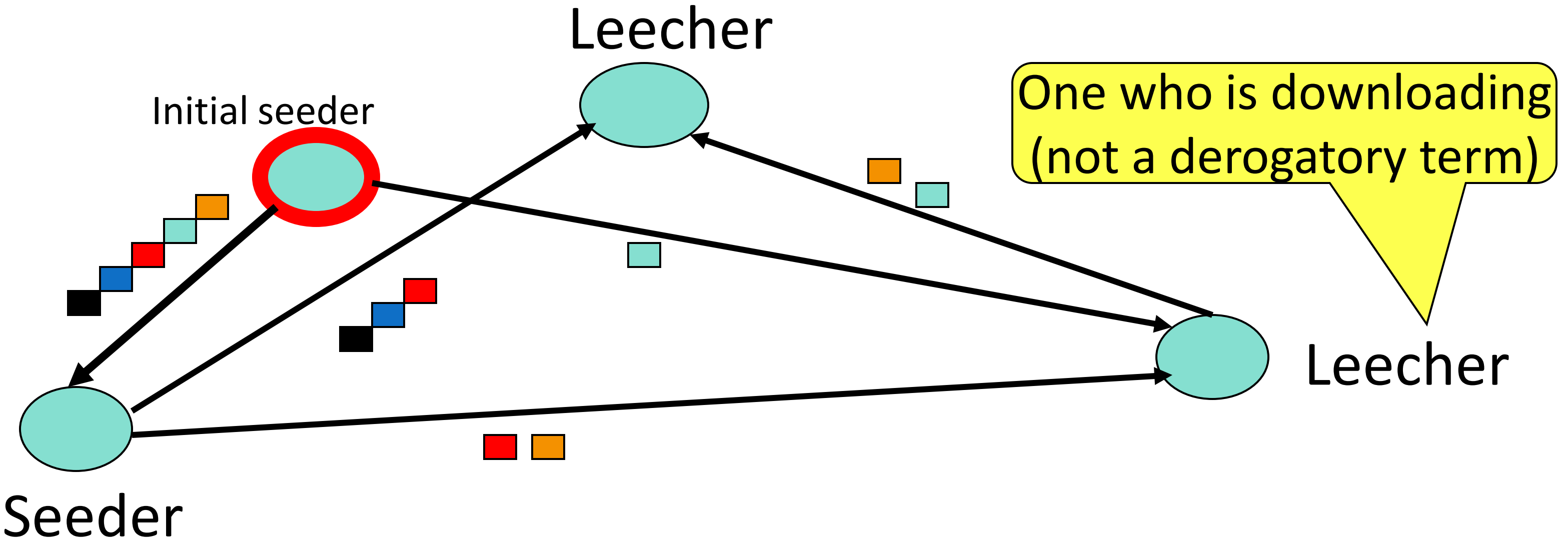
◆ A **swarm** is the set of peers that are participating in distributing the same files

◆ A peer joins a swarm by asking the tracker for a peer list and connects to those peers.

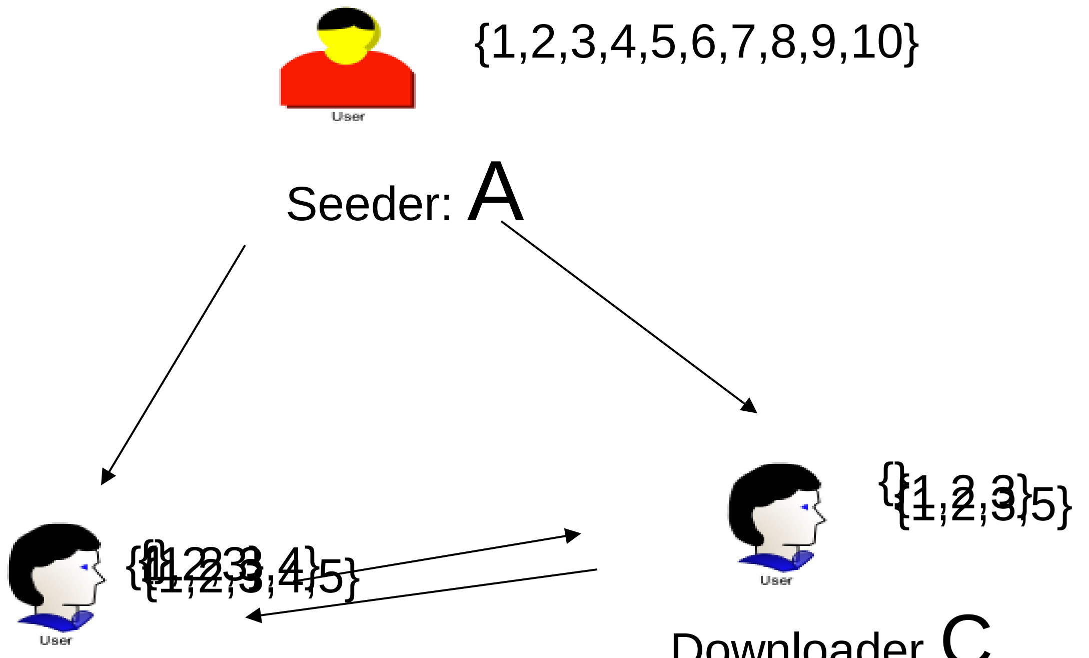
BitTorrent naming conventions

Seeder = a peer that provides the complete file.

Initial seeder = a peer that provides the initial copy.



Simple example



{1,2,3,4,5,6,7,8,9,10}

Seeder: A

{}
{1,2,3}
{1,2,3,4}
{1,2,3,4,5}

{}
{1,2,3}
{1,2,3,5}

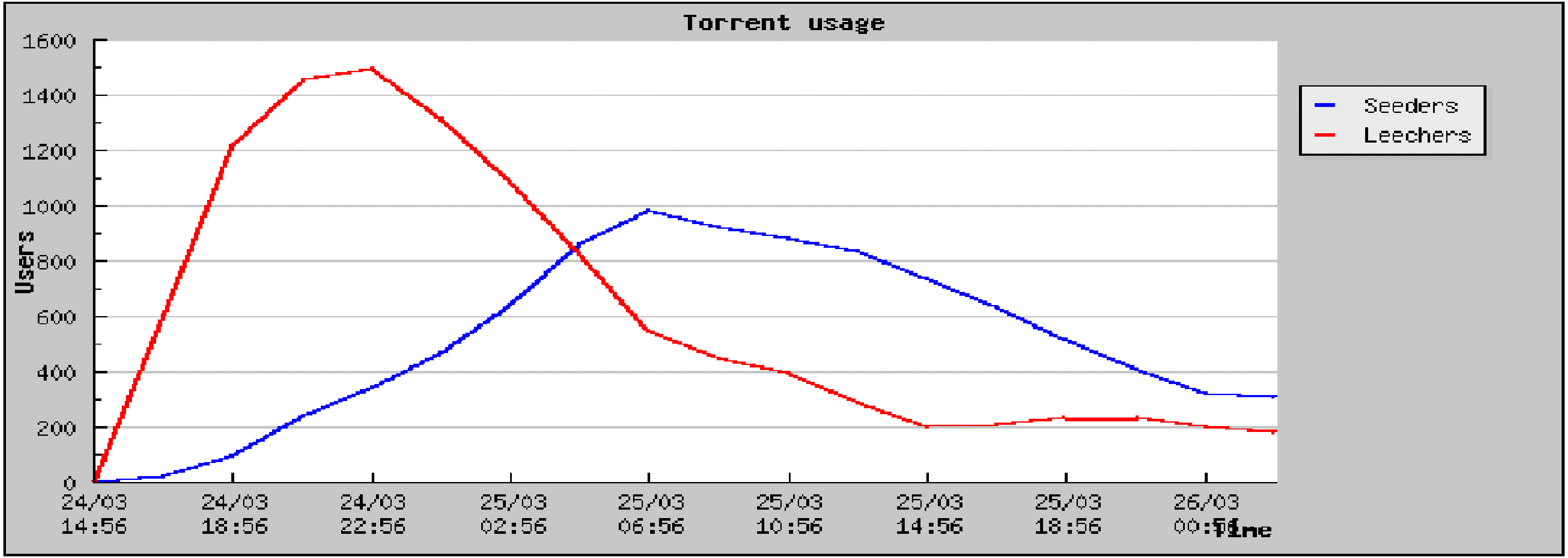
Downloader B

Downloader C

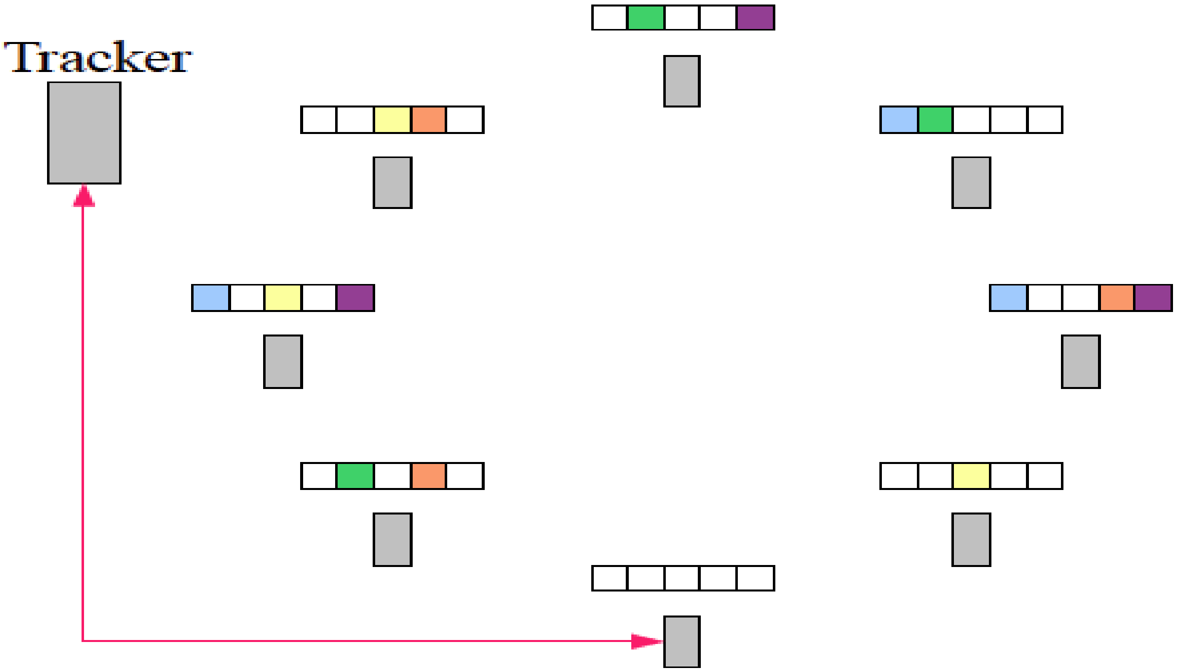
Basic Idea

- As a leecher downloads pieces of the file, replicas of the pieces are created. ***More downloads mean more replicas available***
- As soon as a leecher has a complete piece, it can potentially share it with other downloaders. Eventually each leecher becomes a seeder by obtaining all the pieces, and assembles the file.
- Each leacher
  - Reports to its peers what pieces it has
  - Starts exchanging these pieces with them
- Torrent file on web server has SHA1 hashes of all the pieces
- Peers don’t report that they have a piece until they’ve checked its hash
  - *Could have used erasure codes*

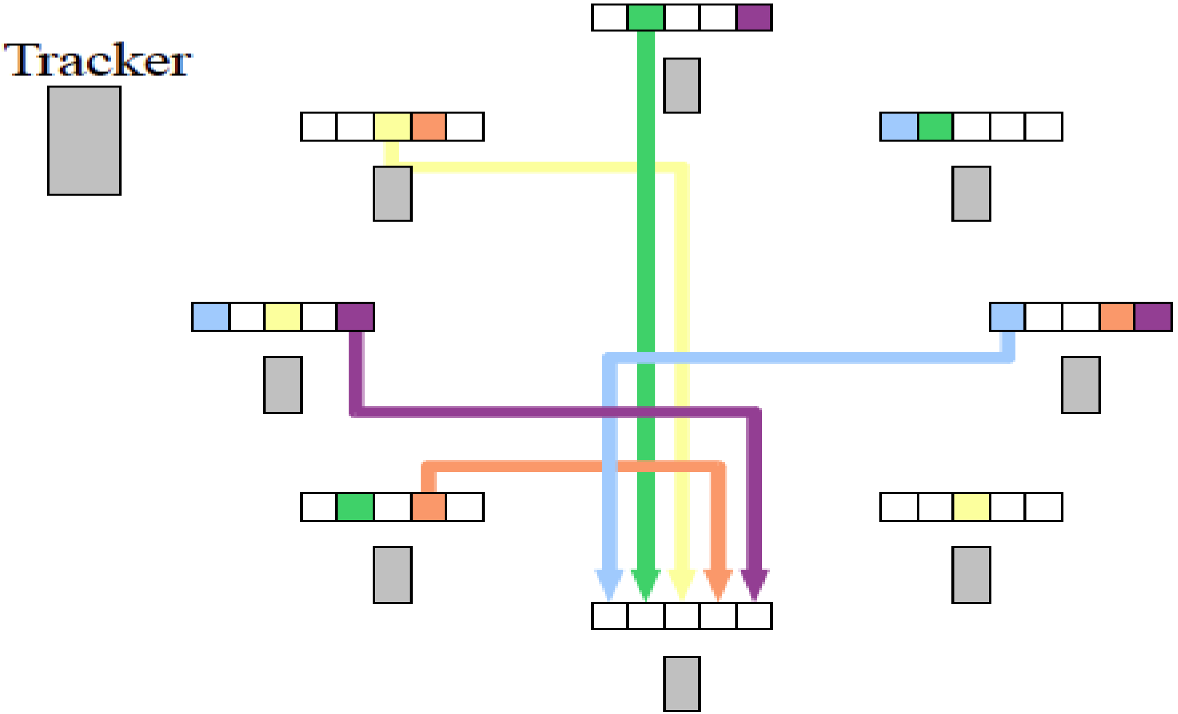
Operation


Torrent usage
Users
1600
1400
1200
1000
800
600
400
200
0
24/03 14:56
24/03 18:56
24/03 22:56
25/03 02:56
25/03 06:56
25/03 10:56
25/03 14:56
25/03 18:56
26/03 00:56
Time
Seeders
Leechers

Download in progress



Download in progress



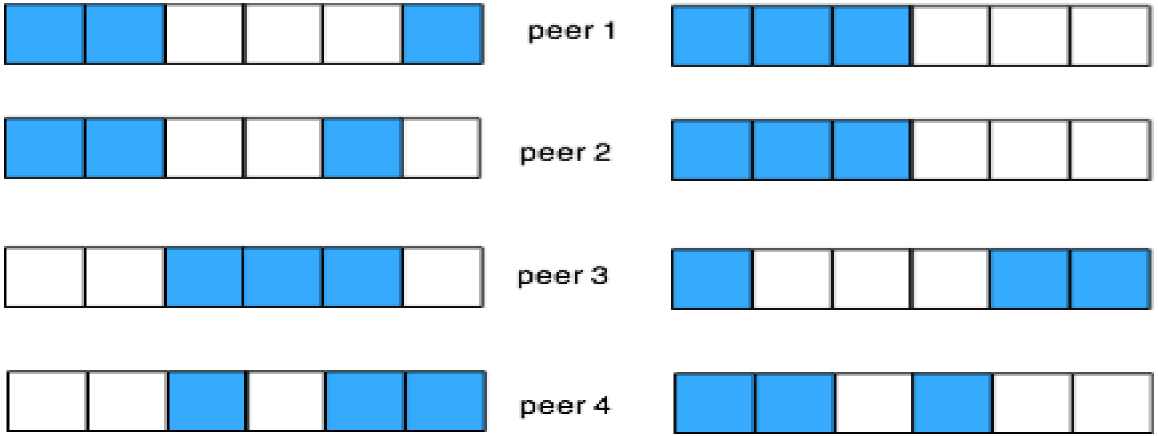
Pipelining

- When transferring data over TCP, always have several requests pending at once (typically 5), to avoid a delay between pieces being sent.
- Every time a piece or a sub-piece arrives, a new request is sent out.

Piece Selection

- The order in which pieces are selected by different peers is critical for good performance
- If an inefficient policy is used, then peers may end up in a situation where each has all identical set of easily available pieces, and none of the missing ones.
- If the original seed is prematurely taken down, then the file cannot be completely downloaded! What are "good policies?"

Piece Selection



Small overlap is good

Large overlap is bad
-- wastes bandwidth

Piece selection

- **Strict Priority**
- **Rarest First**
    - General rule
- **Random First Piece**
    - Special case, at the beginning
- **Endgame Mode**
    - Special case

Random First Piece

- Initially, a peer has nothing to trade
- Important to get a complete piece ASAP
- Select a random piece of the file and download it

Rarest Piece First

- Determine the pieces that are most rare among your peers, and download those first.
  - Increases diversity in the pieces downloaded
    - avoids case where a node and each of its peers have exactly the same pieces; increases throughput
  - Increases likelihood all pieces still available even if original seed leaves before any one node has downloaded entire file

Endgame Mode

- Near the end, missing pieces are requested from every peer containing them.
- This ensures that a download is not prevented from completion due to a single peer with a slow transfer rate.
- Some bandwidth is wasted, but in practice, this is not too much.

BT: internal mechanism

- Built-in **incentive** mechanism (where all the magic happens):
  - Choking Algorithm
  - Optimistic Unchoking

Choking

- **Choking** is a *temporary refusal* to upload. It is one of BT’s most powerful idea to deal with **free riders (those who only download but never upload)**.
- ***Tit-for-tat strategy*** is based on game-theoretic concepts.

Choking

Reasons for choking:

- Avoid free riders
- Network congestion

A good choking algorithm caps the number of simultaneous uploads for good TCP performance.

More on Choking

Peers try out **unused connections** once in a while to find out if they might be better than the current ones (optimistic unchoking).

Optimistic unchoking

- A BT peer has a single “optimistic unchoke” to which it uploads regardless of the current download rate from it. This peer rotates every 30s

- Reasons:
  - To discover currently unused connections that are better than the ones being used
  - To provide minimal service to new peers

Anti-snubbing

- A peer is said to be snubbed if each of its peers chokes it
- To handle this, snubbed peer stops uploading to its peers

➢Optimistic unchoking done more often

  - Hope is that will discover a new peer that will upload to us

Upload-Only mode

- Once download is complete, a peer can only upload. The question is, which nodes to upload to?
- Policy: Upload to those with the best upload rate. This ensures that pieces get replicated faster, and new seeders are created fast

.torrent

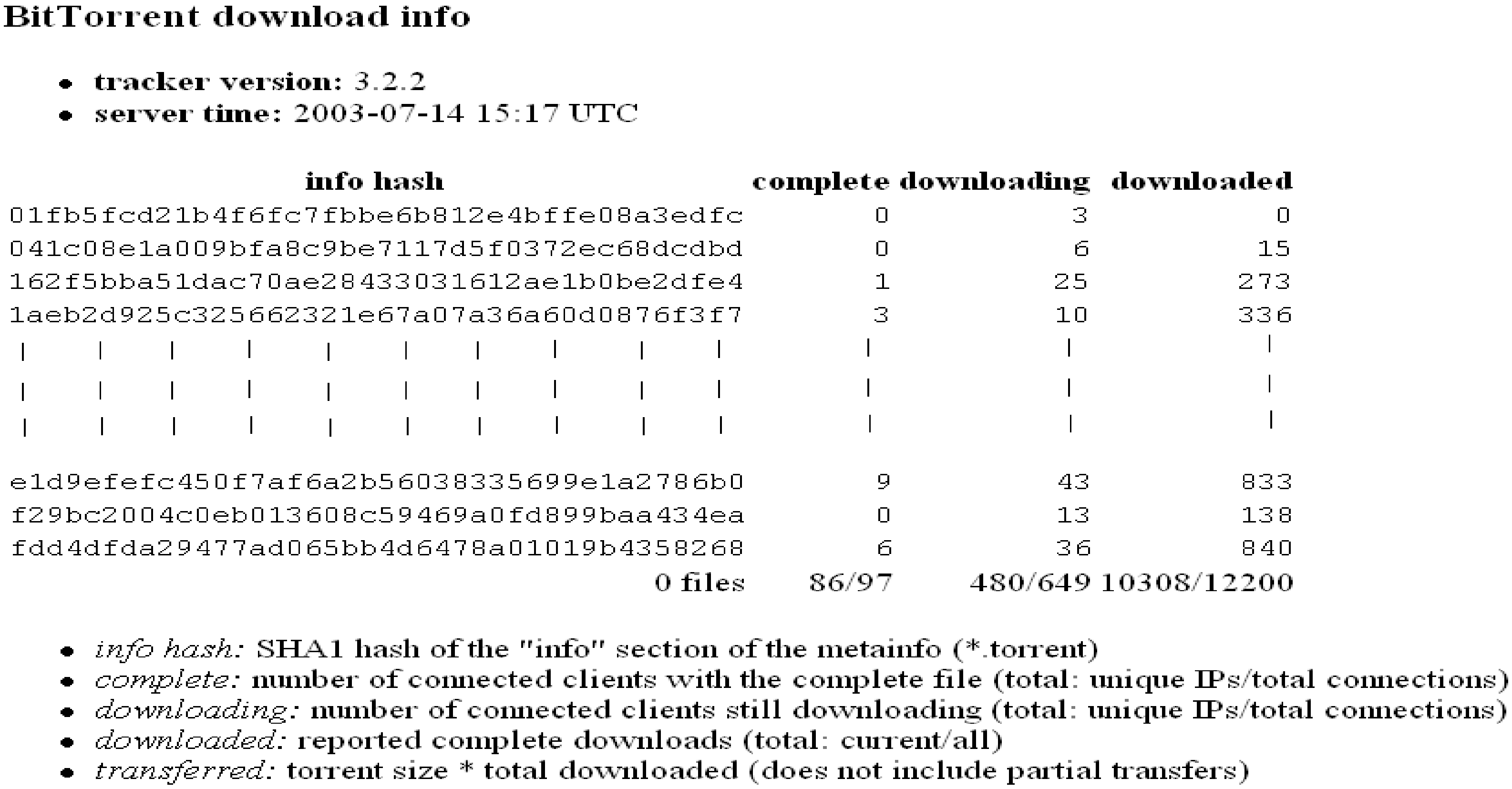
- url of the tracker
- Pieces <hash1,hash2,....hashn>
- Piece length
  - pieces maps to a string whose length is a multiple of 20. It is to be subdivided into strings of length 20, each of which is the SHA1 hash of the piece at the corresponding index.
- Name
- Length
- Files
  - Path
  - length

Bencode

- All data passed in BT is bencoded (BEE – Encoded).
- This is a representation convention used to avoid interoperability problems.

Tracker

- Peer cache
  - IP, port, peer id
- State information
  - Completed
  - Downloading
- Returns random list



BitTorrent download info

- **tracker version:** 3.2.2
- **server time:** 2003-07-14 15:17 UTC

info hash	complete	downloading	downloaded
01fb5fcd21b4f6fc7fbbe6b812e4bffe08a3edfc	0	3	0
041c08e1a009bfa8c9be7117d5f0372ec68dcdbd	0	6	15
162f5bba51dac70ae28433031612ae1b0be2dfe4	1	25	273
1aeb2d925c325662321e67a07a36a60d0876f3f7	3	10	336
| | | | | | | | | |	|	|	|
| | | | | | | | | |	|	|	|
| | | | | | | | | |	|	|	|
e1d9efefc450f7af6a2b56038335699e1a2786b0	9	43	833
f29bc2004c0eb013608c59469a0fd899baa434ea	0	13	138
fdd4dfda29477ad065bb4d6478a01019b4358268	6	36	840
0 files	86/97	480/649	10308/12200

- *info hash:* SHA1 hash of the "info" section of the metainfo (*.torrent)
- *complete:* number of connected clients with the complete file (total: unique IPs/total connections)
- *downloading:* number of connected clients still downloading (total: unique IPs/total connections)
- *downloaded:* reported complete downloads (total: current/all)
- *transferred:* torrent size * total downloaded (does not include partial transfers)

Peer > Tracker:
GET requests has following keys

- info_hash – hash of the .torrent.
- peer_id – My unique ID.
- My IP / Port
- uploaded / Downloaded
- left – Can not be calculated from Downloaded, because of errors / restart.
- event - Why I do the GET.

GET events

- Reasons for calling the GET:
    - Started
    - Completed
    - Stopped
    - Empty - done at regular intervals

Tracker > Peer : tracker GET responses

- failure
- interval – next time to GET
- List of peers

Peer <> Peer communication

Actual packet structure

Peers protocol

- operates over TCP
- Peer connections are symmetrical
- Refers to pieces by index from .torrent
- Connections contain : choked and interested

Peer messages

- Handshake
- Keep alive (0 size, every 2 minutes)
- 0 - choke
- 1 - unchoke
- 2 - interested
- 3 - not interested
- 4 - have
- 5 - bitfield
- 6 - request
- 7 - piece
- 8 - cancel

Change connection state

- 'choke', 'unchoke', 'interested', and 'not interested' have no payload.
- Data transfer takes place:
  - one side is interested, other side is not choking
  - state must be kept up to date at all times
- This is a precondition for upload / download – REQUEST / PIECE

BITFIELD

- Only ever sent as the first message.
- bitfield - a bit for each piece –
  - 1 – Have
  - 0 – Don’t have
- May skip if has 0 pieces.

HAVE

- Single number, the index which that downloader just completed and checked the hash of.

REQUEST

- Contain - index, begin, and length
- Length is generally a power of two unless it gets truncated by the end of the file.

Why BitTorrent took off

- Better performance through “pull-based” transfer
  - Slow nodes don’t bog down other nodes
- Allows uploading from hosts that have downloaded parts of a file
  - In common with other end-host based multicast schemes

Why BitTorrent took off

- Practical Reasons (perhaps more important!)
  - Working implementation (Bram Cohen) with simple well-defined interfaces for plugging in new content
  - Many recent competitors got sued / shut down
    - Napster, Kazaa
  - Doesn’t do “search” per se. Users use well-known, trusted sources to locate content
    - Avoids the pollution problem, where garbage is passed off as authentic content

Pros and cons of BitTorrent

- Pros
  - Proficient in utilizing partially downloaded files
  - Discourages “freeloading”
    - By rewarding fastest uploaders
  - Encourages diversity through “rarest-first”
    - Extends lifetime of swarm
- Works well for “hot content”

Pros and cons of BitTorrent

- Cons
  - Assumes all interested peers active at same time; performance deteriorates if swarm "cools off"
  - Even worse: no trackers for obscure content

Pros and cons of BitTorrent

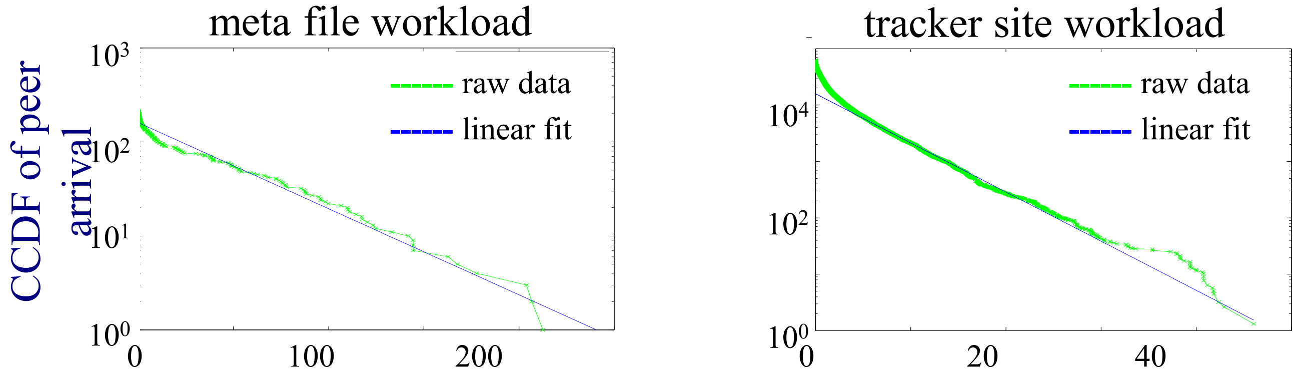
- Dependence on centralized tracker: pro/con?
  - ☹ Single point of failure: New nodes can’t enter swarm if tracker goes down
  - Lack of a search feature
    - ☺ Prevents pollution attacks
    - ☹ Users need to resort to out-of-band search: well known torrent-hosting sites / plain old web-search

“Trackerless” BitTorrent

- To be more precise, “BitTorrent without a centralized-tracker”
- E.g.: Azureus
- Uses a Distributed Hash Table (Kademlia DHT)
- Tracker run by a normal end-host (not a web-server anymore)
  - The original seeder could itself be the tracker
  - Or have a node in the DHT randomly picked to act as the tracker

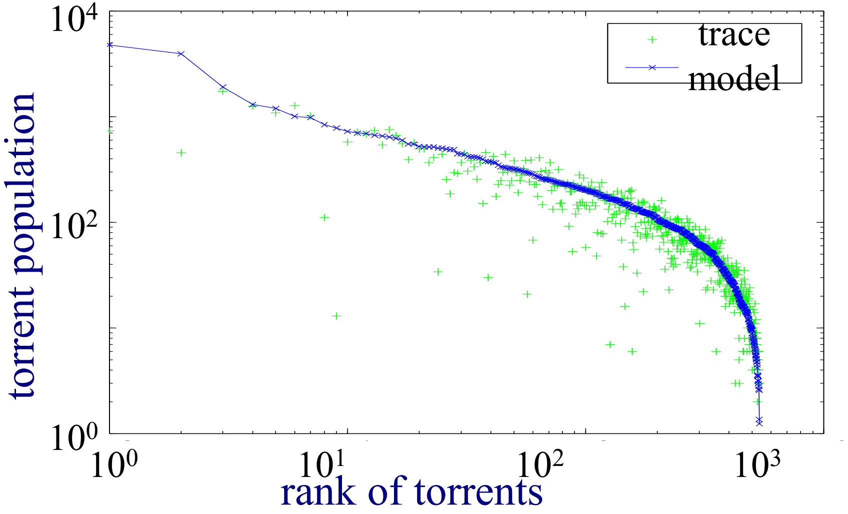
Dynamics of a torrent

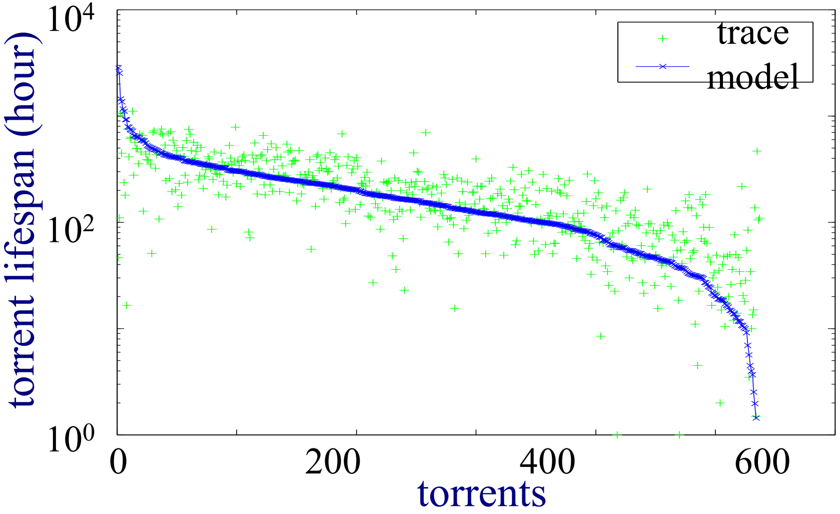
Peer arrival rate $= \dfrac{\text{number of arrivals}}{\Delta t}$



Peer arrivals: decrease with time exponentially

Torrent Population and Lifespan

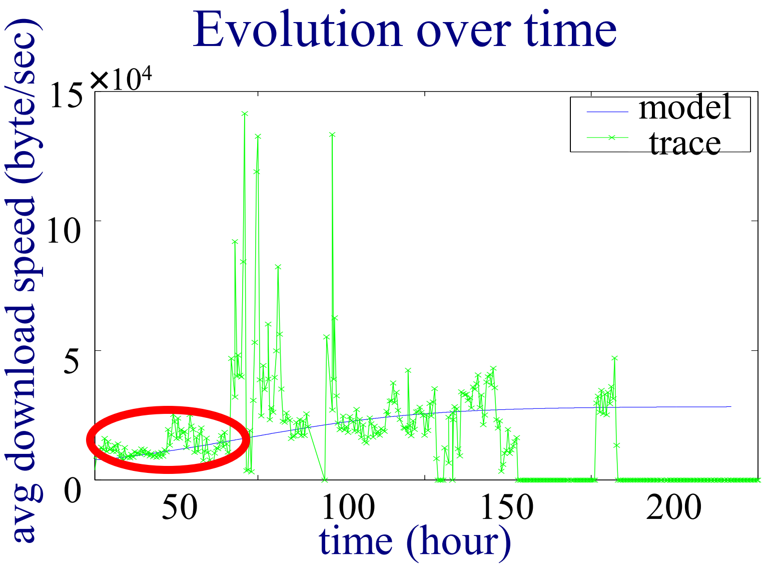


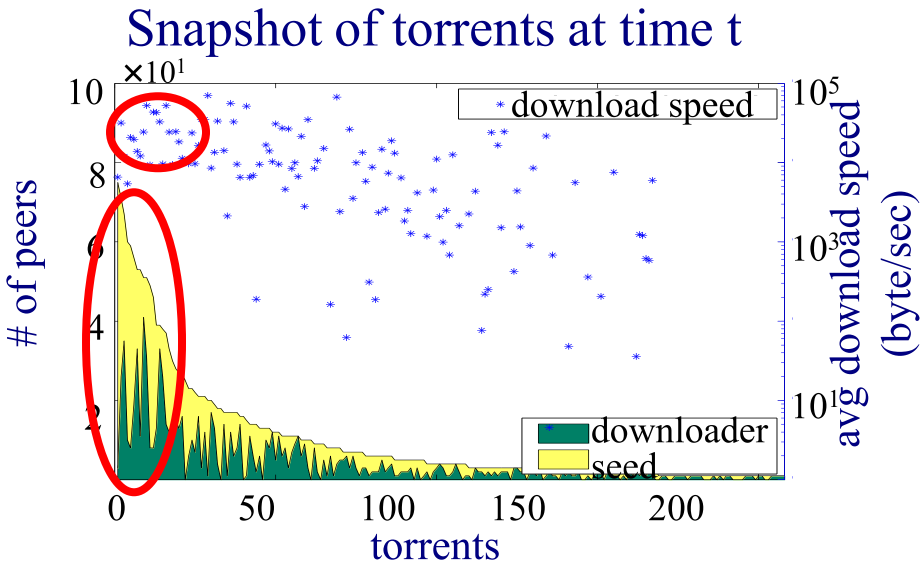


Most torrents are small (avg 102)

Most torrents are short live (avg 8 days)

Performance Stability



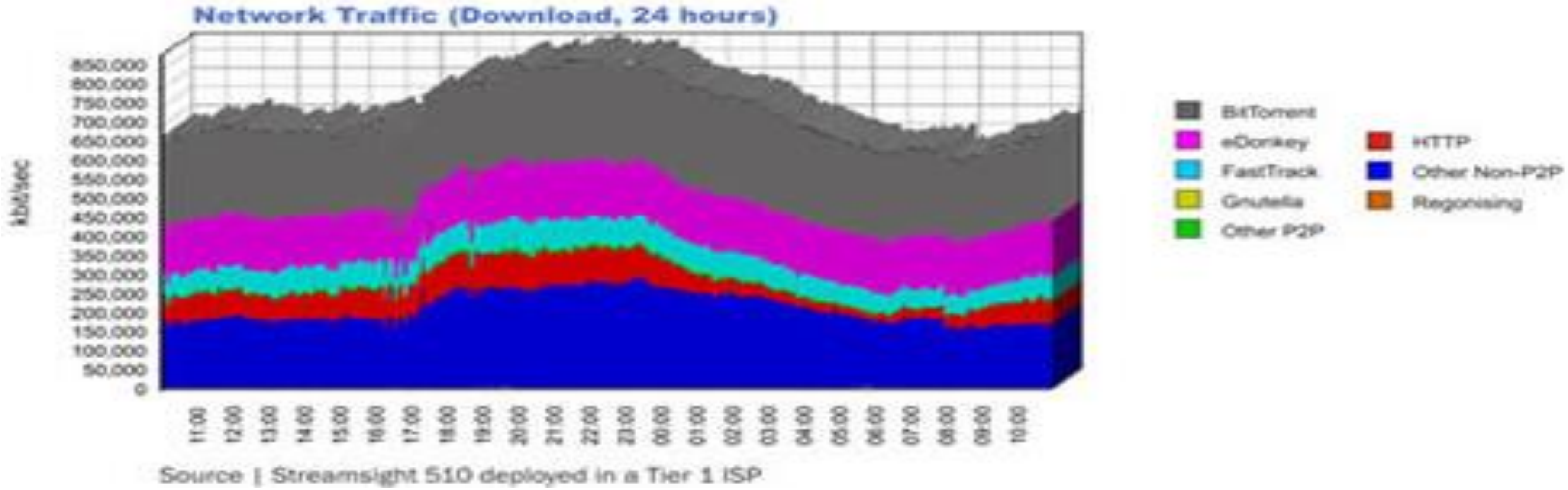


Only stable when torrent is large

Fluctuate significantly after peak time

Larger torrents have higher and more stable performance

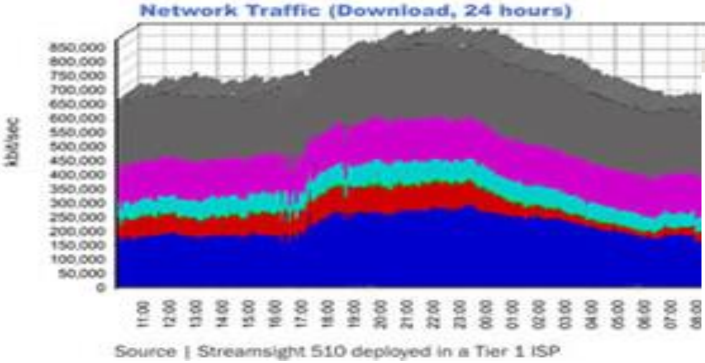
Why is (studying) BitTorrent important?



Source | Streamsight 510 deployed in a Tier 1 ISP

(From CacheLogic, 2004)

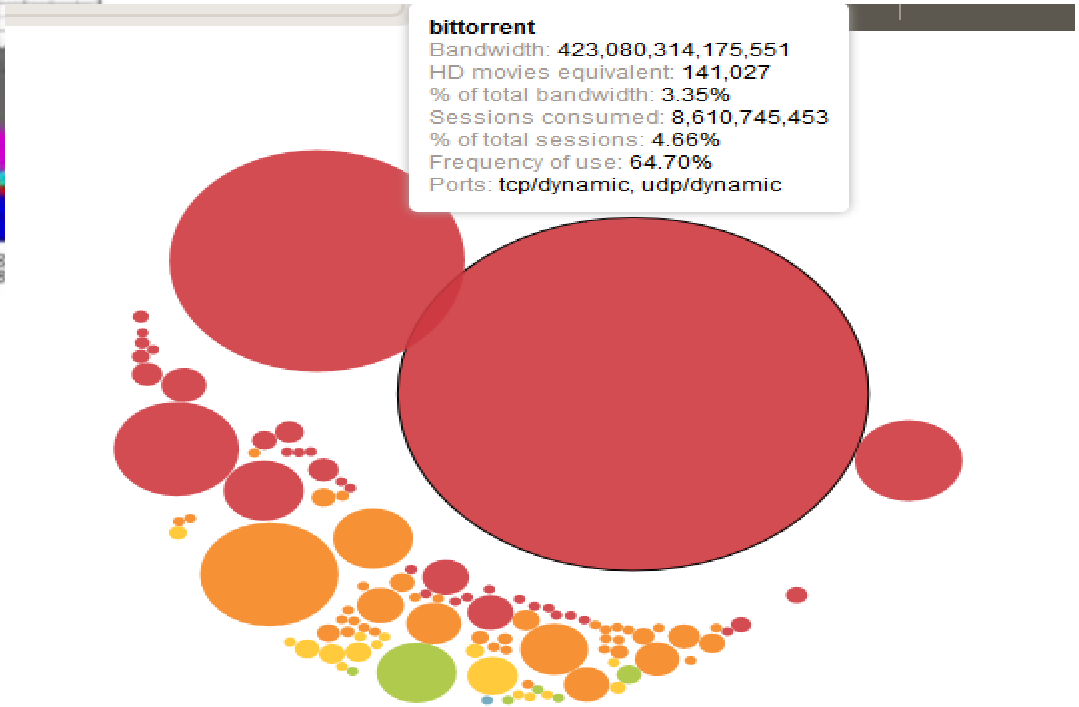
Why is (studying) BitTorrent important?



Network Traffic (Download, 24 hours)

Source | Streamsight 510 deployed in a Tier 1 ISP

(From CacheLogic, 2004)



bittorrent
Bandwidth: 423,080,314,175,551
HD movies equivalent: 141,027
% of total bandwidth: 3.35%
Sessions consumed: 8,610,745,453
% of total sessions: 4.66%
Frequency of use: 64.70%
Ports: tcp/dynamic, udp/dynamic

(From PaloAltoNetworks.com, 2013)

Why is (studying) BitTorrent important?

- BitTorrent consumes significant amount of internet traffic today
  - In 2004, BitTorrent accounted for 30% of all internet traffic (Total P2P was 60%), according to CacheLogic
  - Slightly lower share in 2005 (possibly because of legal action), but still significant
  - BT always used for legal software (linux iso) distribution too
  - Recently: legal media downloads (Fox)

Questions about BT

- What is the effect of bandwidth constraints?
- Is the Rarest First policy really necessary?
- Must nodes perform seeding after downloading is complete?
- How serious is the Last Piece Problem?
- Does the incentive mechanism affect the performance much?

Trackerless torrents

BitTorrent also supports "trackerless" torrents, featuring a DHT implementation that allows the client to download torrents that have been created without using a BitTorrent tracker.

BitTorrent links

- Hivatalos oldal
  - http://bittorrent.com
- FAQ
  - http://www.dessent.net/btfaq/#what
- Torrent-ek
  - http://www.suprnova.org/
    - 2004 végén bezárták
  - http://isohunt.com/
    - 159.000 tracker, 6.8 millió aktív torrent, 161 millió fájl, 12.2 PB adat, 25.8 millió peer

Pirate parties

Pirate Party is a label adopted by political parties in different countries.

Pirate parties support

- civil rights,
- direct democracy and participation in government,
- reform of copyright and patent law,
- free sharing of knowledge (open content),
- information privacy,
- transparency,
- freedom of information,
- anti-corruption
- Internet neutrality
- 7.1% at the Swedish EU elections

The Pirate Bay

- http://thepiratebay.org
- Az egyik legnépszerűbb weboldal a neten
  - 2003 novemberében indult
- 2006. május 31-én a svéd rendőrség lefoglalja a szervereket, 3 napig offline a weboldal
- A per
  - 2009. április 17-én a szolgáltatás működtetőit (Peter Sunde, Fredrik Neij, Gottfrid Svartholm és Carl Lundström) 1 év börtönre és 30 millió SEK (~ 700 millió HUF) büntetésre ítélik
  - Fellebbezés, a bírót elfogultsággal vádolják
- > 25 millió peer (2008 nov.)
- 4 millió regisztrált (2009 dec.) felhasználó
  - A letöltéshez nem kell regisztrálni, csak a kommentekhez és a feltöltéshez