Incentives Build Robustness in BitTorrent

Bram Cohen

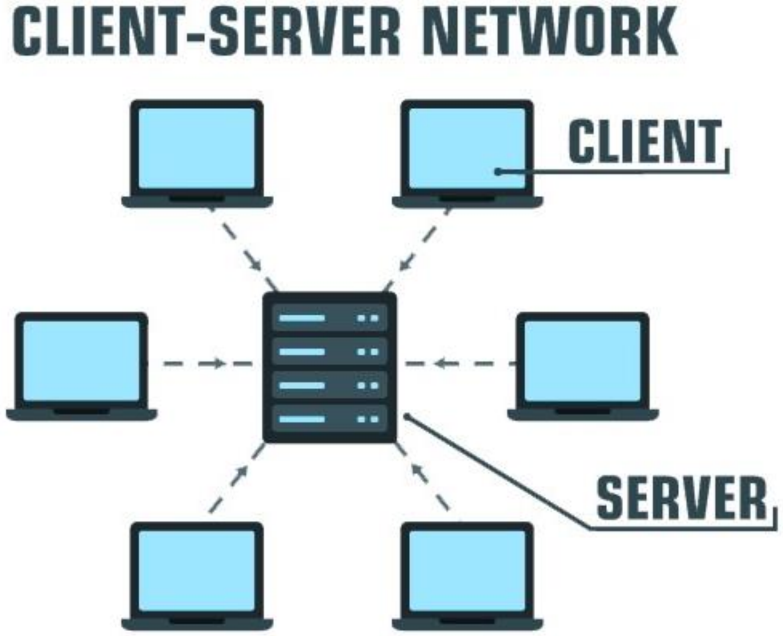
Presenter
Md **Bulbul** Sharif

Outline

- About BitTorrent
- Technical Framework
- Choking Algorithms
- Real World Experience
- Personal Opinions

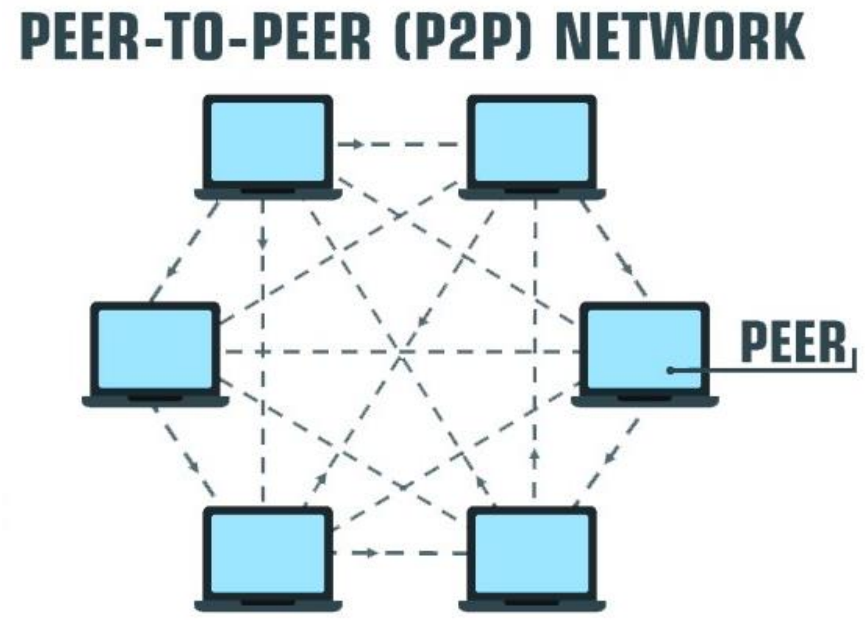
Traditional File Hosting

- Scalability issues for client/server systems
- Server's workload grows linearly with number of clients



BitTorrent Strategy

- Chop file into many pieces
- As soon as a peer has a complete piece, it can trade it with other peers
- Hopefully, we will be able to assemble the entire file at the end



Advantages

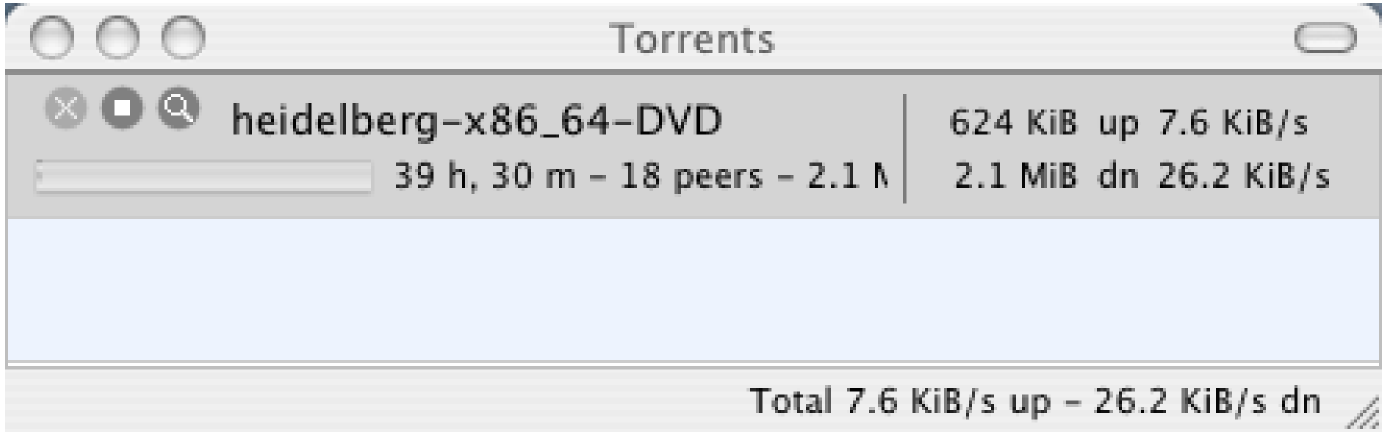
- P2P solutions are
  - Scalable:
    - Downloading bandwidth grows with number of peers
  - Easy to deploy:
    - No additional hardware
    - No change to network infrastructure
  - Cheap

Issues

- Organizing data transfers:
  - Figuring which peers have which chunks of data
  - Deciding where to send these chunks
- Dealing with churning:
  - Peers come and go
- Enforcing fairness:
  - Some peers do not upload as many data as they download

Interface

- Standard “Save As” dialog box



heidelberg-x86_64-DVD

39 h, 30 m – 18 peers – 2.1 M

624 KiB up 7.6 KiB/s

2.1 MiB dn 26.2 KiB/s

Total 7.6 KiB/s up – 26.2 KiB/s dn

Components

- Seed
  - Peer that has the entire file
- Leacher
  - Peer that has an incomplete copy of the file
- Torrent file
  - Contains file info, name, hash info, url for a tracker
- Tracker
  - Centralized process, keeps track of peers
  - Does not distribute actual contents of file, only holds metadata

Deployment

- Decision to use BitTorrent is made by publisher of file
- Users join BitTorrent to get a file they want
  - Most users stops uploading once they have downloaded the file
  - Standard implementation keeps uploading until the BT window closes
- In a typical deployment
  - Number of downloaders having parts of the file (**leeches**) increases very fast then peaks at a maximum before decreasing exponentially
  - Number of downloaders having the whole file (**seeds**) increases more slowly then peaks at a maximum before decreasing exponentially

Technical Framework

- Publishing Content
- Peer Distribution
- Pipelining
- Piece Selection

Publishing Content

- Protocol layered on top of HTTP
- Publisher provides a `.torrent` file
- To start a seed is needed

Peer Distribution

- Tracker helps peers find each other.
- BitTorrent cuts the files into pieces
  - Typically 256KB
  - Uses SHA1 hash to verify integrity
- Peers download from any other peer
  - Peers may not have what is wanted
  - Peers may not allow them to download

Pipelining

- BitTorrent uses TCP
  - Avoid TCP “slow start”
  - Have many pending requests
- Pieces get broken down
  - Sub-pieces typically 16KB in size
  - Try and keep some number (typically 5) of requests pending at a time

Piece Selection

- Strict Priority
  - Finish first downloading pieces of which downloader has one or more sub-pieces
  - Gets complete pieces as quickly as possible
- Rarest First
  - Download first the pieces that the fewest of their own peers have
  - Ensures that peers have the pieces that most of their peers want

Piece Selection (continue…)

- Random First Piece
  - New peer should get its first complete piece as quickly as possible
  - Rare pieces can be downloaded from fewer peers than other pieces
  - New peer will select first pieces to download at random until it has obtained a complete piece
- Endgame Mode
  - At end of download
    - Peer will send to all other peers requests for sub-pieces it doesn’t have yet from all other
    - Will send cancels for all sub-pieces which arrive
  - Objective is to speed up end of download

Choking Algorithms

Choking is a temporary refusal to upload; downloading occurs as normal

- Pareto Efficiency
- BitTorrent's Choking Algorithm
- Optimistic Unchoking
- Anti-snubbing
- Upload Only

Pareto Efficiency

- Pareto efficient system refers to there cant be two counterparties that can make an exchange and both be hampered
- Local optimization -> global optima

BitTorrent's Choking Algorithm

- Penalizes peers that do not reciprocate
  - Tit-for-tat policy
- Peer always unchoke a fixed number of peers (Default 4)
- Unchoke based on current download rate
- Every ten seconds, each peer selects four less cooperating peers it will choke
  - Will refuse to upload data to these peers for ten seconds
  - Long enough for TCP to reach full capacity with the new transfers

Optimistic Unchoking

- Used to discover if a currently choked peer would be better.
- Done every third unchoke decision (30 seconds).

Anti-snubbing

- A peer might be sometimes choked by all peers from which it was downloading
- After 60 seconds without a single piece, the client will assume it has been ‘snubbed’
- Does not upload to a peer that has snubbed it (except as an optimistic unchoke)

Upload Only

- Once download is complete, a peer has no download rates to use for comparison nor has any need to use them
- The question is, which nodes to upload to?
- Upload to those with the best upload rate.
- This ensures that pieces get replicated faster
- Also, peers that have good upload rates are probably not being served by others

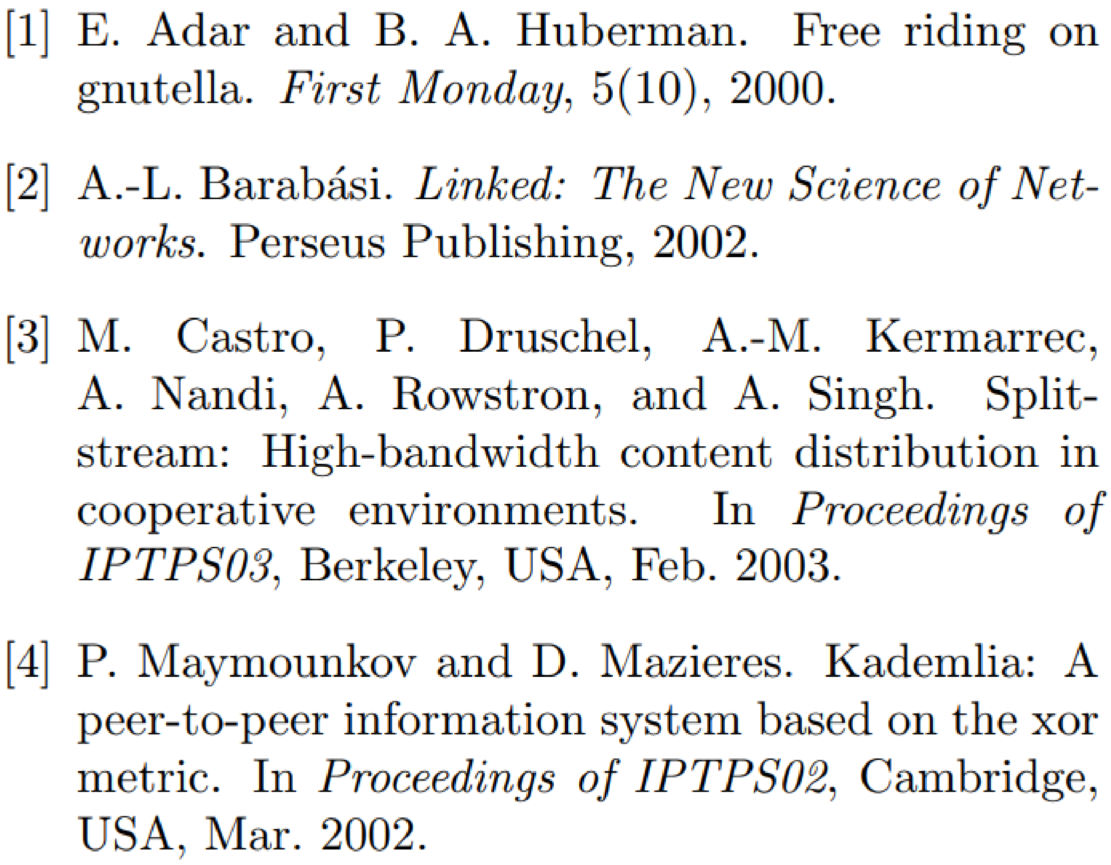
Experience

- BT routinely serves
  - Files 400 megabytes in size
  - 1000 of successful downloads
- Can have over a thousand concurrent downloaders.
- Sole scaling bottleneck appears to be the bandwidth overhead of the tracker

Personal Opinions

- Well written
- Limited references
- All logistical problems of file downloading are handled in the interactions between peers. (**Peer Distribution**)
- Well known economic theories show that systems which are pareto efficient … (**Pareto Efficiency**)
- Built for transferring large files
    - Copyright violations (movies)
- Download without upload or upload less (**BitTyrant**)

Reference



[1] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), 2000.

[2] A.-L. Barabási. *Linked: The New Science of Networks*. Perseus Publishing, 2002.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in cooperative environments. In *Proceedings of IPTPS03*, Berkeley, USA, Feb. 2003.

[4] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS02*, Cambridge, USA, Mar. 2002.

Thanks for Listening

Any Question?