P2P Content Distribution
BitTorrent and Spotify

Amir H. Payberah
amir@sics.se

Amirkabir University of Technology
(Tehran Polytechnic)
Possible Solutions for Content Distribution
Client-Server Model
P.R.O.B.L.E.M.S.
The Client-Server Model Problems

▶ Scalability?
The Client-Server Model Problems

- Scalability?
- Single Point of failure?
The Client-Server Model Problems

- Scalability?
- Single Point of failure?
The Client-Server Model Problem
Scalable and Fault-Tolerant Client-Server Model
Peer-to-Peer Model
Peer-to-Peer (P2P) Model
P2P Challenges

- Churn in the system
- Free-riding problem
- Bottleneck in the overlay network
- Connectivity problem, e.g., NAT
How To Discover Data?

Where is FamilyGuy.avi?

Ubuntu.iso
Pallet.mp3
rakhsh.sics.se

127.mp3
FamilyGuy.avi
castor.sics.se
Central directory

- Ubuntu.iso → rakhsh.sics.se
- Pallet.mp3 → rakhsh.sics.se
- FamilyGuy.avi → castor.sics.se
- 127.mp3 → castor.sics.se
- Ubuntu.iso → rakhsh.sics.se, x.kth.se
Possible Solutions - Second Generation

- Flooding
Possible Solutions - Third Generation

- Distributed Hash Table (DHT)
P2P Content Distribution Applications - Media Streaming

Spotify

PPTV

Joost
BitTorrent is a system for efficient and scalable replication of large amounts of static data.
Files are broken into pieces of size between 64KB and 1MB.
.torrent Files

- Metadata

- Contains:
  - URL of tracker
  - Information about the file, e.g., filename, length, ...
The Core Idea

- A peer obtains the **.torrent** file.
The Core Idea

- A peer obtains .torrent file.
- It, then, connects to the tracker.
The Core Idea

- A peer obtains *.torrent* file.
- It, then, connects to the tracker.
- The tracker tells the peers from which other peers to download the pieces of the file.
The Core Idea

- A peer obtains .torrent file.
- It, then, connects to the tracker.
- The tracker tells the peers from which other peers to download the pieces of the file.
- Peers use this information to communicate with each other.
The Core Idea

- A peer obtains .torrent file.
- It, then, connects to the tracker.
- The tracker tells the peers from which other peers to download the pieces of the file.
- Peers use this information to communicate with each other.
- The peers send information about the file and themselves to tracker.
What About Free Riders?
Tit-For-Tat
Question

- From **which peers** **download** the **pieces**?
Peer Selection

- Use **choking algorithm** to choose peer to download pieces.

- Decision to **choke/unchoke** based on **tit-for-tat**.
Discover More Cooperating Peers

- Optimistic unchoking
- Allocate an upload slot to a randomly chosen uncooperative peer
Snubbed Peers

- If all its peers *choke* it.

- Increase the number of *optimistic unchokes*. 
Which piece?
Rarest first: common parts left for later
Piece Selection

- **Rarest first**: common parts left for later
- **Random first piece**: start-up need to get a complete piece
\textbf{Piece Selection}

- **Rarest first**: common parts left for later
- **Random first piece**: start-up need to get a complete piece
- **Endgame mode**: broadcast for all remaining blocks
BitTorrent Extension

- Distributed tracker
- Peer-exchange
Spotify

- Active users: over 50 million
- Number of songs: over 20 million
- Number of songs added per day: over 20000
- Number of playlists: over 1.5 billion created so far
- Available in 58 countries
- Legal
The Core Idea

- Request first piece from Spotify servers.
The Core Idea

- Request **first piece** from Spotify **servers**.
- Meanwhile, search **P2P** network for remainder.
The Core Idea

- Request **first piece** from Spotify **servers**.

- Meanwhile, search **P2P** network for remainder.

- Switch back and forth between Spotify **servers and peers** as needed.
The Core Idea

- Request **first piece** from Spotify **servers**.

- Meanwhile, search **P2P** network for remainder.

- Switch back and forth between Spotify **servers and peers** as needed.

- Towards **end of a track**, start **prefetching the next one**.
Main Problem in Using Spotify P2P Network

Peer Discovery
Peer Discovery

- Sever-side tracker *(BitTorrent style)*
  - Only remembers 20 peers per track.
  - Returns 10 (online) peers to client on query.
 Peer Discovery

- **Sever-side tracker** (*BitTorrent style*)
  - Only remembers 20 peers per track.
  - Returns 10 (online) peers to client on query.

- **Broadcast** query in small (2 hops) neighborhood in overlay (*Gnutella style*)
Peer Discovery

- **Sever-side tracker** *(BitTorrent style)*
  - Only remembers 20 peers per track.
  - Returns 10 (online) peers to client on query.

- **Broadcast** query in small (2 hops) neighborhood in overlay *(Gnutella style)*

- **LAN peer discovery**
- Ask for most *urgent pieces first*. 
▶ Ask for most *urgent pieces first.*

▶ If a peer is *slow*, re-request from new peers.
Downloading in P2P

- Ask for most **urgent pieces** first.

- If a peer is **slow**, re-request from new peers.

- When buffers are **low**, download from **central server** as well.
Downloading in P2P

- Ask for most urgent pieces first.

- If a peer is slow, re-request from new peers.

- When buffers are low, download from central server as well.

- If buffers are very low, stop uploading.
Spotify vs. BitTorrent

- One (well, three) P2P overlay for all tracks (not per-torrent).
Spotify vs. BitTorrent

- One (well, three) P2P overlay for all tracks (not per-torrent).
- Does not inform peers about downloaded blocks.
Spotify vs. BitTorrent

- One (well, three) P2P overlay for all tracks (not per-torrent).
- Does not inform peers about downloaded blocks.
- Downloads blocks in order.
Spotify vs. BitTorrent

- One (well, three) P2P overlay for all tracks (not per-torrent).
- Does not inform peers about downloaded blocks.
- Downloads blocks in order.
- Does not enforce fairness (such as tit-for-tat).
Spotify vs. BitTorrent

- One (well, three) P2P overlay for all tracks (not per-torrent).
- Does not inform peers about downloaded blocks.
- Downloads blocks in order.
- Does not enforce fairness (such as tit-for-tat).
- Informs peers about urgency of request.
Caching

- Player **caches tracks** it has played.

- Use **10%** of free space (capped at **10GB**)

- **Least Recently Used** policy for **cache eviction**.

- Over **50%** of data comes from **local cache**.
Spotify Data Usage

Data source - ratio - by week

- **Server**
  - Cur: 10.86
  - Min: 6.76
  - Avg: 9.62

- **P2P**
  - Cur: 33.90
  - Min: 23.78
  - Avg: 33.86

- **Cache**
  - Cur: 55.24
  - Min: 48.47
  - Avg: 56.53
Spotify Says Goodbye to P2P

KEEP CALM
IT'S A SHUTDOWN
Summary
Summary

Challenges in P2P Content Distribution:

1. Financial Incentives
2. Quality of Service
3. Security and Privacy

Examples:

- BitTorrent
- Spotify
Questions?