Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming

Gunnar Kreitz    Fredrik Niemelä

KTH – Royal Institute of Technology
Spotify
gkreitz@kth.se

P2P’10, August 27 2010
What is Spotify?

- Peer-assisted on-demand music streaming
- Large catalog of music (over 8 million tracks)
- Available in 7 European countries, over 7 million users
- Fast (median playback latency of 265 ms)
- Legal
User Perspective

- Client software
- Ad-funded (and free), or monthly subscription
- Not included in evaluation data:
  - Can also play local music files (introduced later)
  - Smartphone clients (no P2P)
Comparison with Video-on-Demand

- Lower bitrate
- Shorter objects
- More objects
- Different access pattern
  - More active users
  - Users play their favorite tracks often
Overview of Spotify Protocol

- Proprietary protocol
- Designed for on-demand streaming
- 96–320 kbps audio streams (most are Ogg Vorbis q5, 160 kbps)
- Relatively simple and straightforward design
Caches

- Player caches tracks it has played
- Default policy is to use 10% of free space (capped at 10 GB)
- Caches are large (56% are over 5 GB)
- Least Recently Used policy for cache eviction
- Over 50% of data comes from local cache
- Cached files are served in P2P overlay
Streaming a Track

- Request first piece from Spotify servers
- Meanwhile, search for peers with track
- Download data in-order
- When buffers are sufficient, only download from P2P
- Towards end of a track, start prefetching next one
Playout-buffer Adjustment

- Minimize latency while avoiding stutter
- TCP throughput varies
  - Sensitive to packet loss
  - Bandwidth over wireless mediums vary
- Model throughput as a Markov chain and simulate
- Heuristics
Security Through Obscurity

▶ Client must be able to access music data
▶ Reverse engineers should not be able to access music data
▶ So, some details are secret (and the client is obfuscated)

Image by XKCD [http://xkcd.com/730/], CC BY NC 2.5
P2P Structure

- Unstructured overlay (not a Distributed Hash Table)
- Nodes have fixed maximum degree (60)
- Neighbor eviction by heuristic evaluation of utility
- No overlay routing
- Looks for and connects new peers when streaming new track
- Overlay becomes (weakly) clustered by interest
Brief Comparison to BitTorrent

- One (well, two) 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
Finding Peers

- Server-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)
- Client uses both mechanisms for every track
Evaluation

- So, how well does it work?
- Collected measurements 23–29 March 2010
Data Sources

Data source - ratio - by week

Weekday
Morning
Weekend
Night

Cur: 10.86 6.76 9.62
Min: 33.90 23.78 33.86
Avg: 55.24 48.47 56.53

Server
P2P
Cache
Data Sources

- Mostly minor variations over time
  - Better P2P performance on weekends
  - P2P most effective at peak hours
- 8.8% from servers
- 35.8% from P2P
- 55.4% from caches
Latency and Stutter

- Median latency: 265 ms
- 75th percentile: 515 ms
- 90th percentile: 1047 ms
- Below 1% of playbacks had stutter occurrences
Track Accesses

- There is no cost per track for users
- What does the usage pattern look like?
- How is that affected by caches and P2P?
Track Accesses

(a) Track playback frequencies (normalized), log-log scale
(b) Track server request frequencies (normalized), log-log scale

Figure: Frequency of track accesses

- 60% of catalog was accessed
- 88% of track playbacks were within most popular 12%
- 79% of server requests were within the most popular 21%
Finding Peers

Table: Sources of peers

<table>
<thead>
<tr>
<th>Sources for peers</th>
<th>Fraction of searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracker and P2P</td>
<td>75.1%</td>
</tr>
<tr>
<td>Only Tracker</td>
<td>9.0%</td>
</tr>
<tr>
<td>Only P2P</td>
<td>7.0%</td>
</tr>
<tr>
<td>No Peers Found</td>
<td>8.9%</td>
</tr>
</tbody>
</table>

- Each mechanism by itself is fairly effective
Protocol Overhead

Table: Distribution of application layer traffic in overlay network

<table>
<thead>
<tr>
<th>Type</th>
<th>Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music Data, Used</td>
<td>94.80%</td>
</tr>
<tr>
<td>Music Data, Unused</td>
<td>2.38%</td>
</tr>
<tr>
<td>Search Overhead</td>
<td>2.33%</td>
</tr>
<tr>
<td>Other Overhead</td>
<td>0.48%</td>
</tr>
</tbody>
</table>

▶ Measured at socket layer
▶ Unused data means it was cancelled/duplicate
Summary

* Commercially deployed system
* Custom protocol for Music-on-demand streaming
* Peer-assisted
Future Problems

- Playout strategy adapted to P2P streaming
- User satisfaction metrics
- Music-on-demand streaming
- Specialized overlays exploiting similarity in taste