Spotify — Behind the Scenes

Gunnar Kreitz

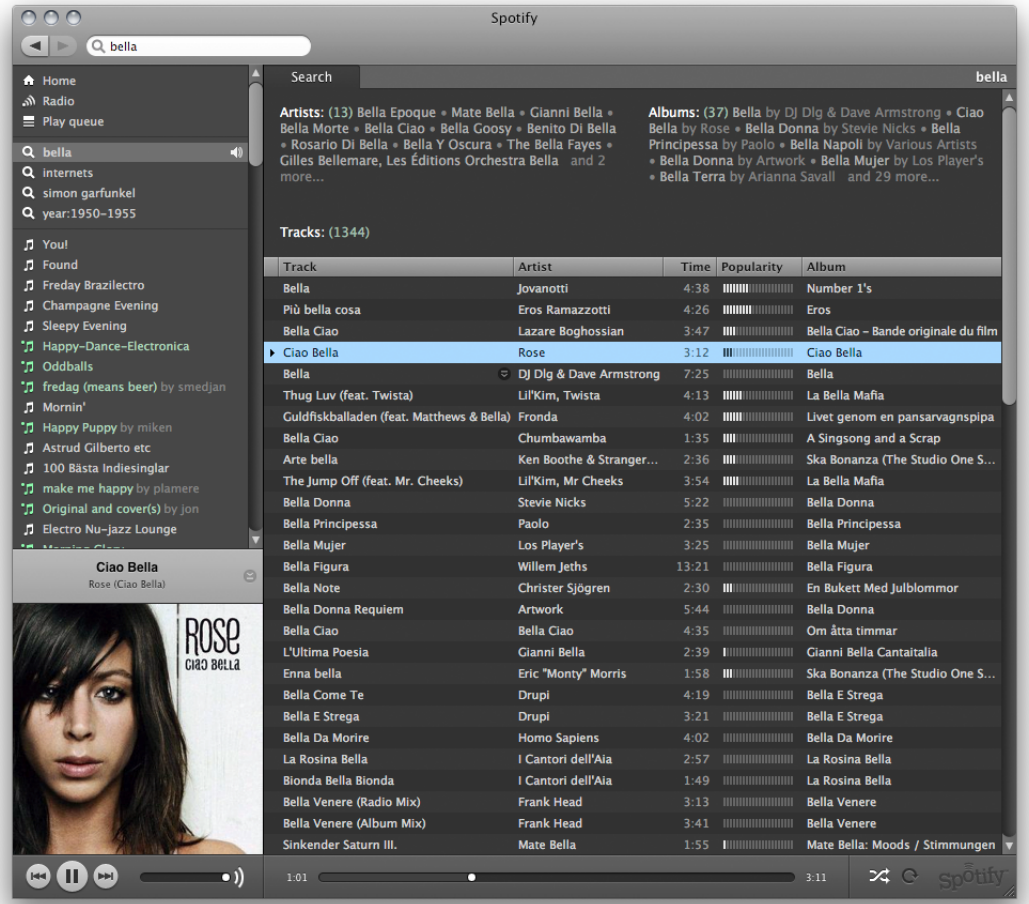
Spotify
gkreitz@spotify.com

SNUS, Apr 20 2010

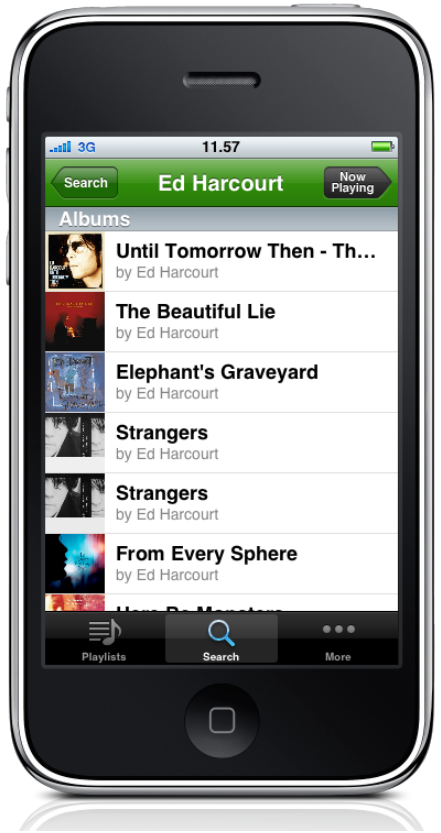
What is Spotify?

- Lightweight on-demand streaming
- Large catalogue of music
- Fast
- Legal

What is Spotify?



What is Spotify?



Business Idea

- More convenient than piracy

Business Idea

- More convenient than piracy
- Free version with ads (invite needed)
  - Daypass gets rid of ads for 24 hours
- Premium version (no invite needed)
  - Mobile
  - Offline mode
  - API access
  - No ads

Technical Overview of Client Software

- Client software on Mac and Windows
  - Works well under Wine
- Client written in C++ with some Objective-C++
- libspotify on Linux
- Desktop clients autoupdate (with signed binaries)

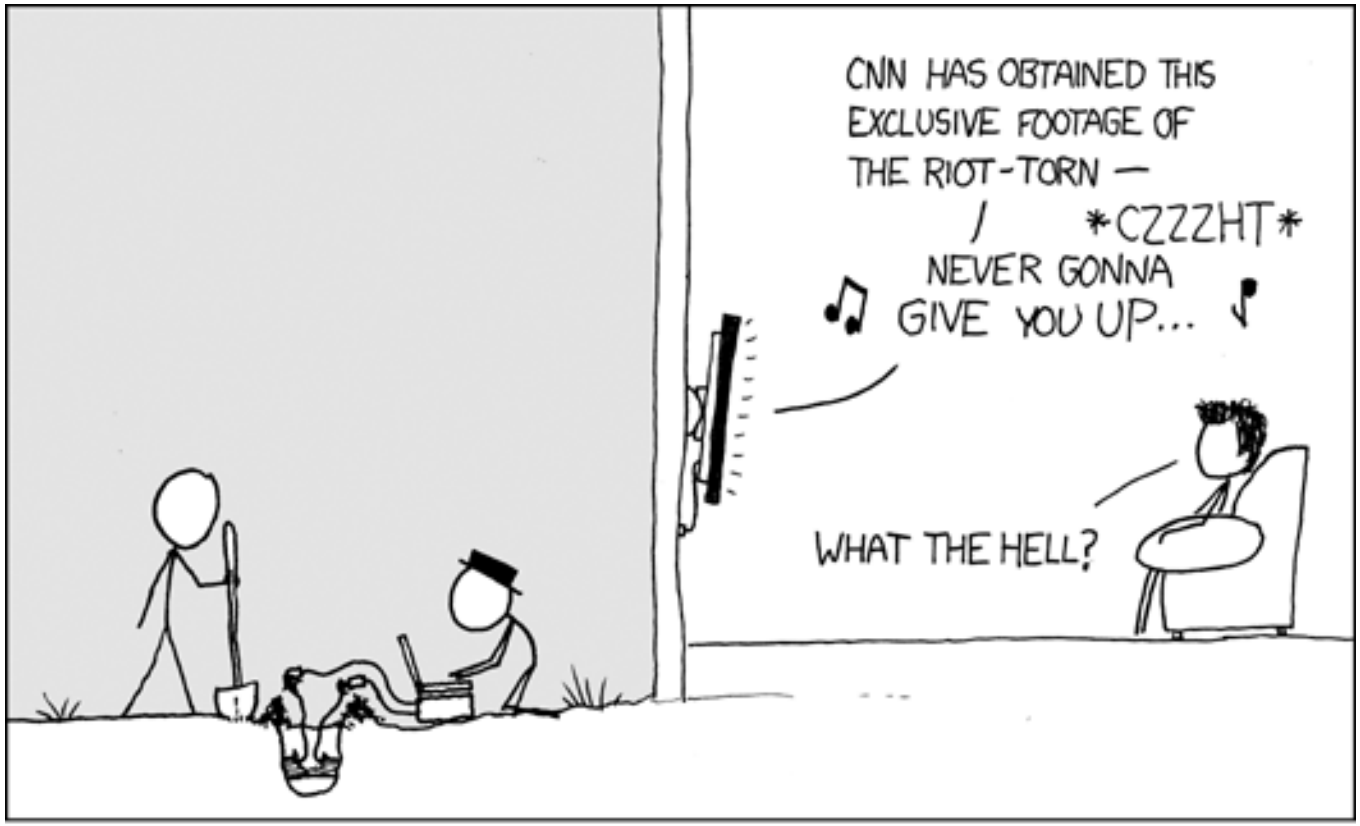
Everything is a link

- spotify: URI scheme
- `spotify:track:6JEK0CvvjDjjMUBFoXShNZ#0:44`
- `spotify:user:gkreitz:playlist:4W5L19AvhsGC3U9xm6lQ9Q`
- `spotify:search:never+gonna+give+you+up`

Everything is a link

- spotify: URI scheme
- `spotify:track:6JEK0CvvjDjjMUBFoXShNZ#0:44`
- `spotify:user:gkreitz:playlist:4W5L19AvhsGC3U9xm6lQ9Q`
- `spotify:search:never+gonna+give+you+up`
- New URI schemes not universally supported
- `http://open.spotify.com/track/6JEK0CvvjDjjMUBFoXShNZ#0:44`

Links contain opaque id:s



GREAT MOMENTS IN TROLLING:
RICK ASTLEY IS SUCCESSFULLY RICKROLLED

(Image from XKCD, `http://www.xkcd.com/351`)

Metadata API

- Simple, http-based API
- Search and lookup
- `http://ws.spotify.com/lookup/1/?uri=spotify:track:6JEK0CvvjDjjMUBFoXShNZ`
- `http://ws.spotify.com/search/1/artist?q=foo`

Overview of Network Protocol

- Proprietary protocol
- 96-320kbps audio streams
- Streaming from
  - Spotify servers
  - Peers
  - Content Delivery Network (http)

Spotify Protocol

- Designed for random-access streaming rather than live streaming
- (Almost) Everything over TCP
- (Almost) Everything encrypted
- Multiplex messages over a single TCP connection
- Persistent TCP connection to server while logged in

Playing a Random Track

- Request first piece from Spotify servers
- Meanwhile, search Peer-to-peer (P2P) for remainder
- Switch back and forth between Spotify servers and peers as needed
- Towards end of a track, start prefetching next one

Latency Kills!

- Latency kills (not just in First Person Shooters)

Latency Kills!

- Latency kills (not just in First Person Shooters)
- Increasing latency of Google searches by 100 to 400ms decreased usage by 0.2% to 0.6% [Brutlag09]
- The decreased usage persists

Latency Kills!

- Latency kills (not just in First Person Shooters)
- Increasing latency of Google searches by 100 to 400ms decreased usage by 0.2% to 0.6% [Brutlag09]
- The decreased usage persists
- Most Spotify playbacks start within a few hundred milliseconds

The forbidden word

(By http://www.flickr.com/photos/marxalot/, CC BY-SA 2.0)

TCP Congestion Window

- TCP maintains several windows, among them `cwnd`
- `cwnd` is used to avoid network congestion
- A TCP sender can never have more than `cwnd` un-ack:ed bytes outstanding
- Additive increase, multiplicative decrease

TCP Congestion Window

- TCP maintains several windows, among them `cwnd`
- `cwnd` is used to avoid network congestion
- A TCP sender can never have more than `cwnd` un-ack:ed bytes outstanding
- Additive increase, multiplicative decrease
- What to do with `cwnd` when a connection sits idle?
- RFC 5681 (TCP Congestion Control) says:

> *Therefore, a TCP SHOULD set cwnd to no more than RW before beginning transmission if the TCP has not sent data in an interval exceeding the retransmission timeout.*

TCP Congestion Window and Spotify

- Spotify traffic is bursty
- Initial burst is very latency-critical
- Want to avoid needless reduction of congestion window
- Configure kernels to not follow the RFC 5681 SHOULD.

When to Start Playing?

- Minimize latency while avoiding stutter
- TCP throughput varies
  - Sensitive to packet loss
  - Bandwidth over wireless mediums vary

When to Start Playing?

- 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

- We want our client to be able to access plaintext music data
- We don't want reverse engineers to be able to access plaintext music data
- So we can't tell you exactly how our client works

Security through obscurity

- We want our client to be able to access plaintext music data
- We don't want reverse engineers to be able to access plaintext music data
- So we can't tell you exactly how our client works
- Plus, we need to apply software obfuscation
- Which is impossible [BGIRSVY01] (but we can still make life painful for reverse engineers)

Security through obscurity

(Image from XKCD, `http://www.xkcd.com/257`)

Peer to Peer (P2P) Goals

- Minimize bandwidth bills
- Minimize hardware costs
- Minimize latency
- Avoid stutter

P2P Structure

- Unstructured network (not a Distributed Hash Table)
- Nodes have fixed maximum degree
- Neighbor eviction by heuristic evaluation of utility
- No overlay routing

P2P Structure

- All peers are equals (no supernodes)
- A user only downloads data she needs
- P2P network becomes (weakly) clustered by interest
- Oblivious to network architecture

Brief comparison to Bittorrent

- Bittorrent fetches random blocks, we don't
- Bittorrent enforces fairness (tit-for-tat), we don't
- Bittorrent informs neighbors of what's downloaded so far, we don't
- We inform peers of urgency of requests
- We have one P2P network for all tracks (not per-torrent)

Finding Peers

- Partial central index (Napster-style)
  - Retains small number of peers per track
- Broadcast query in small neighborhood (Gnutella-style)
- Limited broadcast for local (LAN) peer discovery

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
  - When doing so, estimate what point P2P will catch up from
- If buffers are very low, stop uploading

Limit resource usage

- Cap number of neighbors
- Cap number of simultaneous uploads
  - TCP Congestion Control gives “fairness” between connections
- Cap cache size
- Mobile clients don’t participate in P2P

Security in our P2P Network

- Control access to participate
- Verify integrity of downloaded files
- Data transfered in P2P network is encrypted
- Usernames are not exposed in P2P network, all peers assigned pseudonym

Avoiding hijacking

- A peer cannot ask peers to connect to arbitrary IP adress/port
  - Avoiding DDoS issues
- Misbehaving peers are reported

NAT Traversal

- Asks to open ports via UPnP
- Connection requests routed via central servers
- Attempts connections in both directions
- Rather high connection failure rate
- Room for improvement

Challenges Ahead

- Continued growth
  - Increasing user numbers
  - Increasing geographic distribution
- Mobile clients vs. P2P
- Network locality awareness

References


Brutlag
Speed Matters for Google Web Search


Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang
On the (Im)possibility of Obfuscating Programs (Extended Abstract)