Measurements on the Spotify Peer-Assisted Music-on-Demand Streaming System

Mikael Goldmann
KTH – Royal Institute of Technology, and Spotify
Stockholm, Sweden
Email: migo@kth.se

Gunnar Kreitz
KTH – Royal Institute of Technology, and Spotify
Stockholm, Sweden
Email: gkreitz@kth.se

***Abstract*—Spotify is a streaming service offering low-latency access to a large library of music. Streaming is performed by a combination of client-server access and a peer-to-peer protocol. The service currently has a user base of over 10 million and is available in seven European countries. We provide a background on the Spotify protocol with emphasis on the formation of the peer-to-peer overlay.**

Using measurement data collected over a week by instrumenting Spotify clients, we analyze general network properties such as the correspondence between individual user accounts and the number of IP addresses they connect from and the prevalence of Network Address Translation devices (NATs). We also discuss the performance of one of the two peer discovery mechanisms used by Spotify.

I. Introduction

Spotify is a peer-assisted music streaming service. That is, it uses client-server streaming, but clients also participate in a peer-to-peer overlay in order to offload the servers. The service has a library of over 13 million tracks, allowing users to freely choose tracks they wish to listen to and to seek within tracks. The service launched in October 2008 and now has over 10 million users in Finland, France, Great Britain, the Netherlands, Norway, Spain, and Sweden.

In this paper, we report on measurement data collected in the Spotify service. Spotify clients contain instrumentation code which reports various measurement data to the back end. This allows us to perform a large-scale measurement study on properties of user's connection. A study giving more details on the overall structure of the Spotify protocol, together with measurement data on its properties was presented by Kreitz and Niemelä [1].

During our measurement period, the service was offered in four versions: two free versions with advertisements, and two pay-per-month versions. The original free version allowed unlimited streaming on computers running either Windows or OS X, and an invite was required to sign up. Later a free version that did not require an invite was added, with streaming limited to twenty hours per month.[1] Both pay-per-month versions of the service allow unlimited streaming, without ads. The higher priced premium version includes extra features such as the option to stream at a higher bitrate, to synchronize tracks for offline usage, and stream to mobile devices.

The music catalog is the same for free and paying users with the exception of some pre-releases exclusive to premium users. However, due to licensing restrictions, which tracks that are available to a user depends on the user's home country.

A. Related Services

There are several on-demand music streaming services offered today. To our knowledge, all such services but Spotify are web-based, using either Adobe Flash or a web browser plug-in for streaming. Furthermore, they are pure client-server applications with no peer-to-peer component. Among the more well-known such services are Napster, Rhapsody, and We7.

There also exist a number of music locker services today. Amazon and Google have both released such services recently (March and May 2011), and Apple has recently announced plans to launch a similar service. Locker services differ from on-demand streaming services in that a user can only access content she has uploaded and/or purchased. To the best of our knowledge, all current music locker services are pure client-server applications.

The application of peer-to-peer techniques to streaming is more prevalent when it comes to video-on-demand services, like PPLive, PPStream, and Voddler. These vary between supporting live streaming, video-on-demand access, or both. While there are many similarities between video-on-demand and music-on-demand streaming, there are also many differences; including user behavior, the size of streaming objects, and the number of objects offered for streaming. A more detailed discussion is given in [2].

B. Our Contribution

We analyze data collected from a large-scale peer-assisted system. The data is collected by Spotify's servers, providing us details not typically available to an outside observer of a peer-to-peer network, such as accurate user identities in the form of user names and countries. We investigate the performance of the peer-to-peer part of Spotify's system. We also consider more general network questions, such as whether the notion of "one IP, one user" appears to hold to any degree in spite of techniques such as NAT and DHCP. The data collected by the Spotify clients is not specifically tailored to this latter

[1] At the time of writing, the free version allowing unlimited streaming is no longer available, but the traffic data used in this article is from March 2011, before any changes to the free service were announced.

investigation, so we cannot come to as detailed conclusions as [3] (see below), but we can show that there are noticeable differences between different countries and provide plausible reasons for why this is so.

C. Related Work

There have been several studies measuring the performance and behavior of large peer-to-peer systems, describing and measuring both on-demand streaming [4], and file-sharing protocols [5], [6]. Huang *et al.* [7] describe the PPLive video-on-demand streaming system, and also present measurements on its performance.

A study by Casado and Freedman investigates prevalence of NAT devices and larger proxies [3]. For their experiment, measurement code in the form of JavaScript or a Java applet was added to a number of web pages, as well as to a fraction of traffic in a Content Distribution Network (CDN).

II. Spotify Overview

The Spotify protocol is a proprietary network protocol designed for streaming music. There are clients for Linux, OS X, and Windows as well as for several smartphone platforms and hardware music players such as Logitech Squeezebox, Onkyo, and Sonos. The smartphone clients and hardware devices do not participate at all in the peer-to-peer protocol, but only use client-server streaming. Thus, in the remainder of this paper, we only discuss the function of, and data collected from, the OS X and Windows clients.[2]

The clients are closed-source software available for free download, but to use a client, a Spotify user account is needed. Clients automatically update themselves, and only the most recent version is allowed to access the service.

The user interface is similar to those found in desktop mp3 players. Users can organize tracks into playlists. Finding music is organized around two concepts: searching and browsing. A user can search for tracks, albums, or artists, and she can also browse—for instance, when clicking on an artist name, the user is presented with a page displaying all albums featuring that artist.

Spotify is a legal streaming service; all music available for streaming is licensed to Spotify. Users can also play any mp3 files they have on their local drive from the Spotify client, as well as include them in playlists together with streaming tracks. Furthermore, the Spotify client will attempt to "link" local files to tracks in the streaming library by matching the metadata. By default, the client will play the local mp3 file instead of streaming the track when a file has been linked.

Playlists as well as albums, artists, and tracks can be shared with others in the form of URIs. Further encouraging users to share URIs, Spotify clients can directly post URIs and messages to Facebook, Messenger, and Twitter.

Audio streams are encoded using Ogg Vorbis with a default quality of q5, which has variable bitrate averaging roughly 160 kbps. Users with a premium subscription can choose (through a client setting) to instead receive Ogg Vorbis in q9 quality, averaging roughly 320 kbps. Both types of files are served from both servers and the peer-to-peer network. No re-encoding is done by peers, so a peer with the q9 version of a track cannot serve it to one wanting the q5 version.

[2]We exclude Linux as the current Linux client is a preview version only available to paying subscribers.

A. Connection Behavior

While UDP is the most common transport protocol in streaming applications, Spotify instead uses TCP. Firstly, having a reliable transport protocol simplifies protocol design and implementation. Secondly, TCP is friendly to the network in that TCP's congestion control is friendly to itself (and thus other applications using TCP), and the explicit connection signaling assists stateful firewalls in better maintaining their state tables. Thirdly, as streamed material is shared in the peer-to-peer network, the re-sending of lost packets is useful to the application.

While a client is running, it keeps open a TCP connection to a Spotify server. Application layer messages are buffered, and sorted by priority before being sent to the operating system's TCP buffers. For instance, messages needed to support interactive browsing are prioritized over bulk traffic. Application-layer keep-alive messaging is used to quickly detect if the TCP connection is closed, e.g., because of network issues. If disconnected, the client periodically attempts to reconnect to a server. Disconnections do occur regularly, for instance due to temporary network outages or a laptop going to sleep mode.

Some requests and reporting messages will be re-sent once a connection has been re-established. For instance, if a client was playing a track and was disconnected before it has the entire track, it will request track data as soon as it successfully connects to the service.

Between a pair of clients in the peer-to-peer overlay only a single TCP connection is used, and the application protocol multiplexes messages over the connection. Just like the connection to the Spotify server, application-layer keep-alive messages are used to detect disconnections. In the overlay, however, the client does not automatically attempt to reconnect if a connection is lost.

Clients, upon startup, use the Universal Plug n' Play (UPnP) protocol to ask home routers for a port to use for incoming peer-to-peer connections. When a client wishes to connect to a peer, a request is also forwarded through the Spotify server asking the connectee to attempt a TCP connection back to the connecter. This means that as long as one of the two peers can accept incoming connections, the connection will succeed. Apart from these two mechanisms, Spotify clients do not perform any NAT traversal.

III. Spotify's Peer-to-Peer Network

In this section we describe the mechanisms by which Spotify's overlay network is formed to provide background on its peer discovery mechanisms in particular.

We remark that the peer-to-peer overlay is actually split into two, as the service is run from two data centers, one in Stockholm and one in London. A client selects which data center

to connect to based on the response to a DNS query for the SRV record `_spotify-client._tcp.spotify.com`. The servers and weights in the record depend on in which country a GeoIP database indicates that the resolver's IP is located. During our measurement period, Swedish and Finnish users were directed to the Stockholm site, Norwegian users uniformly randomly selected a site, and all others were directed to the London site. A client can only connect to new peers which are connected to servers at the same data center as itself. If it reconnects to a different site it does not drop its peers, so in rare cases a client may have peers from both sites.

A. General Structure

The peer-to-peer overlay used is an unstructured network, the construction and maintenance of which is assisted by trackers. This allows all peers to participate in the network as equals so there are no "supernodes" performing any special network-maintenance functions.

A client will connect to a new peer when it wishes to download a track it thinks the peer has, and this is the only mechanism by which new connections are added to the overlay. It locates peers likely to have a track it is looking for through the mechanisms described in Section III-B. There is no general routing performed in the overlay network, so two peers wishing to exchange data must be directly connected.

Clients store relatively large local caches of the tracks they have played. This material is also what a client uploads to other peers in the overlay. Clients will only serve complete tracks[3] in the overlay to simplify the protocol and reduce overhead.

B. Locating Peers

Two mechanisms are used to locate peers having content the client is interested in. The first uses a tracker deployed in the Spotify back end, and the second a query in the overlay network.

The Spotify tracker is similar to a BitTorrent tracker. It maintains a mapping from tracks to a short list of peers who have recently reported that they have the track. As a peer only offers to serve a track if it has the whole track cached, peers listed in the tracker have the whole track.

As two complementary mechanisms are used, the implementation of each can be kept fairly simple. In particular, the tracker only keeps a list of the 20 most recent peers for each track. Furthermore, clients only report to the tracker when they play a track, rather than periodically reporting the contents of their caches, or notifying the tracker when a track is evicted from the client cache. As clients keep a TCP connection open to a Spotify server, the tracker knows which clients are currently online. When a client asks the trackers for peers who have a track, the tracker replies with a random selection of up to 10 peers who are currently online (the response is limited in size to minimize overhead).

[3]A track may be partially cached if the user plays parts of it and then skips to another track.

In addition to the tracker-based peer searches, clients also sends search requests in the overlay network, similar to the method used in Gnutella. When searching for a track, a client sends a search request to all its neighbors in the overlay, who forward the request to all their neighbors. Thus, the searcher will find a peer with the track if any peer at distance one or two in the overlay has the track. Search queries sent by clients have a query id associated with them, and peers remember the 50 most recent searches seen, allowing them to ignore duplicate messages. This limited message forwarding is the only overlay routing in the Spotify peer-to-peer protocol.

There are several reasons for why Spotify has implemented dual mechanisms for peer discovery. As mentioned, it allows both mechanisms to be imperfect, and thus cheaper in terms of implementation, overhead, or both. It also provides better availability. If the tracker becomes inoperable, peer discovery through the search questions in the overlay is still possible, or if a client becomes disconnected from the overlay, it can find peers through the tracker. If there was only a single mechanism, any failure in that mechanism would result in the overlay immediately being rendered inoperative. As the Spotify system is peer-assisted, this would mean that the entire load of serving all data would shift to the central server, implying that they either need to be significantly over-provisioned, or become overloaded. Highly published failures such as Skype's outage in 2010 [8] demonstrate that recovering a peer-to-peer network from failure can be difficult and slow.

C. Neighbor Selection

Keeping the state required to maintain a large number of TCP connections to peers can be expensive, in particular for home routers acting as stateful firewalls and NATs. Thus, each client has a maximum number of peers it may be connected to at any given time. Clients are configured with both a soft and a hard limit, and never go above the hard limit. The client does not make new connections above the soft limit (but still accepts incoming connection requests) and periodically prunes its connections to keep itself below the soft limit (with some headroom for new connections). These limits are set to 50 and 60, respectively.

When a client needs to disconnect one or more peers, it performs a heuristic evaluation of the utility of each connected peer. The intention is for the heuristic to take into account both how useful the connection is to the evaluating peer, as well as how useful the link is to the overlay as a whole.

The client sorts all its connected peers according to 6 criteria: bytes sent in the last 10 minutes, bytes sent in the last 60 minutes, bytes received in the last 10 minutes, bytes received in the last 60 minutes, the number of peers found through searches sent over the connection in the last 60 minutes, and the number of tracks the peer has that the client has been interested in in the last 10 minutes. For each criterion, the top scoring peer in that criterion gets a number of points, the second peer a slightly lower number, and so on (with slightly different weights for the different criteria). Peers with a raw score of 0 for a criterion do not get any

Table I
PERCENT OF USERS LOGGING IN FROM k IP ADDRESSES DURING THE MEASUREMENT PERIOD.

k	ES	FI	FR	GB	NL	NO	SE
1	46.27	59.15	53.55	54.12	73.75	51.82	53.35
2	22.69	16.79	25.12	20.72	17.33	25.28	20.30
3	10.73	6.77	8.91	8.70	4.65	10.21	7.92
4	6.10	4.09	4.35	4.83	1.85	4.80	4.41
5	3.90	2.85	2.45	3.06	0.90	2.61	2.99
6	2.65	2.10	1.63	2.12	0.47	1.58	2.25
7	1.90	1.61	1.12	1.51	0.31	1.01	1.79
8	1.32	1.29	0.78	1.10	0.21	0.65	1.36
9	0.94	1.03	0.49	0.79	0.13	0.45	1.06
10	0.69	0.77	0.34	0.59	0.08	0.31	0.83

Table II
PERCENTAGE OF IP ADDRESSES WHICH CONNECTED WITH VARIOUS NAT PROPERTIES DURING MEASUREMENT PERIOD

		UPnP		
Country	No NAT	worked	mismatch	private IP
ES	11.58	9.09	2.13	0.61
FI	60.15	11.54	0.97	1.30
FR	11.00	47.58	10.16	0.98
GB	11.68	38.87	11.59	0.64
NL	14.77	18.04	2.19	3.05
NO	23.73	27.44	5.65	5.21
SE	45.47	26.69	2.06	1.34

points for that criterion. The peers points are then summed over all the criteria, and the peers with the least total scores are disconnected.

IV. SPOTIFY MEASUREMENTS

In this section, we present and discuss measurements illustrating the network properties and behavior of clients connected to the Spotify system.

A. Measurement Methodology

Both Spotify clients and servers perform continuous instrumentation and monitoring of the system. The raw log messages are collected and stored on log servers and in a Hadoop cluster (an open-source map-reduce and distributed storage implementation), where they are available for processing.

We collected instrumentation data for the week beginning Monday 14 March. Times are given in UTC. As Spotify is only available in seven countries in Western Europe, most users are on Central European Time which is one hour ahead of UTC. There are strong periodic patterns (e.g., usage is higher during daytime than in the night) in activity and number of users logged in. These are further described in [1].

Each Spotify user has a registered country. Where we investigate differences between country, we use the country associated with the user account. Users on free accounts can keep using Spotify while traveling for at most 2 weeks. Whether a user is traveling is determined from their IP using a commercial GeoIP database.

B. Correspondence between IPs and Users

There has been a number of works measuring properties of deployed peer-to-peer systems, see [9] for a survey of studies. One interesting metric is the number of users or machines connected in peer-to-peer systems. However, peer-to-peer protocols typically do not provide a way to trace user identities. Commonly, researchers performing measurements then instead use the IP address of peers as an identifier. Spotify implements counter-measures against users sharing accounts. In particular, if a user is logged in with several clients simultaneously, playing music on one pauses all the others. Thus, we believe Spotify user accounts correspond well to actual users.

We have investigated logins over our measurement period, counting the number of IPs a user has logged in from. As a user may log in both from home and from work, we have also looked at logins during typical working hours and typical non-working hours. We defined typical working hours as Monday through Friday 9:00–16:00 UTC, and typical non-working hours as Monday through Friday before 7:00 and after 18:00 as well as the entire Saturday and Sunday.

To our surprise, it turns out that for a given country, the results for working and non-working hours are quite similar. For instance, for Sweden 61.6% of the users that logged in several times during working hours used the same IP each time, and 61.5% of the users logging in more than once during non-working hours used the same IP. If we disregard time of day, the corresponding number for Sweden is 53.3%. As differences between working hours and non-working hours were minor, although not insignificant, for all countries, we focus on the data set covering the entire measurement period. We also compared weekdays to weekends with similar results.

Differences between countries are more noticeable as can be seen in Table I. The two outliers are Spain and the Netherlands. In Spain, it is more common for users to log from multiple IPs, and it is relatively rare for users to always log in from the same IP. We believe one reason for this could be if large ISPs in Spain more aggressively change dynamically allocated IPs through DHCP. In the Netherlands, we find the situation reversed. It is significantly more common among Dutch users to only log in from a single IP. One contributing factor may be that Spotify launched much more recently in the Netherlands than in the other 6 countries, and it may be the case that usage patterns change over time (e.g., users install Spotify on more computers).

C. Prevalence of NATs and UPnP support

In a peer-to-peer network, peers require the ability to connect to each other. This is not always possible due to the presence of NAT devices and firewalls. The presence of a NAT device or firewall can be mitigated either by a client using a protocol such as UPnP or NAT-PMP to open a port for incoming connection, or through various hole punching techniques [10], [11], [12].

Spotify clients implement UPnP and report the success status (and external IP, port if successful) to the server. In addition, they also report the local IP address of the network interface used to connect to the server. This data allows us to investigate the prevalence of NAT devices, as well as how

widely supported UPnP is. We remark that the data we present here is only based on reports from the client, we did not connect to supposedly open ports to verify connectivity. Thus, our data does not allow us to distinguish between those who have a firewall that would still block incoming connections from those who have not.

We consider an address private if it is in one of the 10/8, 169.254/16, 172.16/12, or 192.168/16 ranges. For this experiment we used the public IP address, from which the server saw the client's traffic, as identifier.[4] For each IP we see connecting, we look at the events during our entire measurement period. If there ever was a client with a local interface IP matching the IP seen by Spotify servers, we say that there is no NAT. If this did not happen, but the client reported at least once that it used UPnP for incoming connections and the IP UPnP returned matched the IP seen by Spotify servers, we say that UPnP worked. We also report two other categories, when UPnP returned a different IP[5] and when UPnP returned a private IP. The latter case is indicative of a client being behind multiple NATs. In Table II we present the fraction of IP addresses in each category, broken down by country.

There are some striking differences between the network properties of users from different countries, as can be seen in Table II. While we have not been able to fully investigate the reasons for these, we believe we have some partial explanations. Firstly, it can be seen that a surprisingly large fraction in both Finland and Sweden have public IP addresses. We believe that this may be due to the prevalence of 3G and 4G data subscriptions. These are typically used by consumers as a USB device connected directly to their computer, providing Internet connectivity without a NAT. Secondly, Spain has particularly low success rate with UPnP. We believe that it is likely to be the case that large ISPs in Spain either provide its customers home routers with UPnP disabled.

In our study, we also computed separate statistics for weekdays and weekends for each country. Our hypothesis was that there would be significant differences, as we would not expect work networks to provide UPnP functionality. However, the data showed only minor variations, and for space reasons we do not provide the full results here.

Overall, our statistics show that NAT devices are very prevalent today. In most countries, these would likely present a significant problem for peer-to-peer network connectivity. Implementing support for UPnP to open ports appears important as it significantly increases the number of peers who can accept connections. In France and Great Britain, this measure increases the fraction of peers who can receive connections from just above 10% to well over 50%.

D. Locating Peers

As discussed in Section III-B, Spotify uses two mechanisms to find peers with a track. The overall success rate of these was evaluated in [1], where it was shown that the overlay based mechanism returned answers for 82.1% of queries, and the tracker for 84.1%. We remark that their measurement period was a week, one year prior to ours. In this section we look further at the overlay based peer discovery mechanism to evaluate how successful it is in finding new peers, and how successful it is in discovering that already connected peers have requested material.

We measure distance in the peer-to-peer overlay, and thus say that an already connected peer is at distance 1, and that a peer one hop away in the overlay (a peer of a peer) is at distance 2. Looking at the global statistics of the network as a whole over our measurement period, the overlay based mechanism returned answers for 81.8% of the queries. This is very similar to the value from [1], which is to be expected as there have been no major changes to the overlay in the last year. Globally, for 60.4% of queries, at least one peer was found at distance 1, and for 77.4% of queries, at least one peer at distance 2, so clearly it is worthwhile that a client's peers forward requests to their peers. From these statistics, we see that it is both worthwhile to ask already connected peers (as requesting data from them is quicker than connecting to new peers) and that forwarding the query one step is a good peer discovery mechanism.

Furthermore, we also analyzed differences between countries and days of the week. Here, we saw significant differences in the performance of the peer discovery mechanism between various countries, as well as between weekdays and weekends. We present this data in Table III.

Table III
PERCENTAGE OF REQUESTS FOR WHICH PEERS WERE FOUND AT DISTANCE d IN OVERLAY SEARCH.

Country	Weekend	Any d	$d = 1$	$d = 2$
ES	No	77.6	53.4	73.5
ES	Yes	81.8	56.9	77.9
FI	No	84.0	63.6	79.3
FI	Yes	88.0	68.0	83.5
FR	No	69.0	52.3	59.4
FR	Yes	75.3	57.5	64.9
GB	No	79.0	58.7	73.7
GB	Yes	83.5	63.0	77.8
NL	No	71.2	49.9	65.1
NL	Yes	75.4	53.7	69.0
NO	No	79.6	57.5	75.7
NO	Yes	86.2	63.4	82.5
SE	No	84.3	64.4	80.6
SE	Yes	89.0	69.0	85.4

For all countries, the overlay search performs significantly better during weekends than weekdays. So far, we have not been able to explain this phenomenon. As we discussed in Section IV-C, we have not seen significant differences between weekdays and weekends in terms of NAT prevalence or UPnP support. It may be the case that the difference in overlay performance is attributable to differences in user behavior (e.g., music choice and activity), but it may also be due to some other network property than the prevalence of NATs and UPnP support.

We have also been unable to explain the surprisingly large

[4]We group reports to avoid clients reconnecting often being given disproportionally large weight.

[5]This may be legitimate, or due to a bug in the router or the Spotify client.

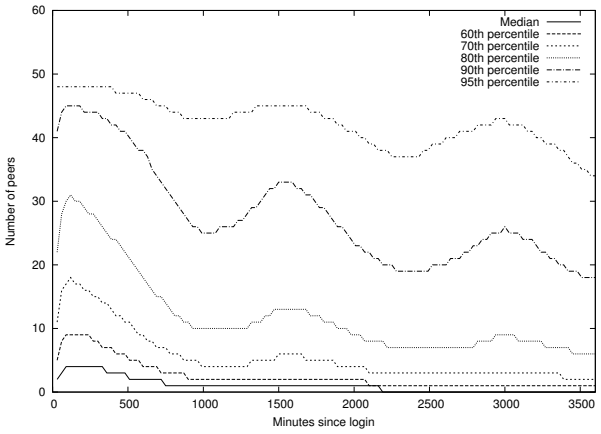

Figure 1. Number of peer connections as a function of time since login.

differences observed between different countries. Again, the presence of NATs and UPnP does not seem to have a strong effect on the success rate of using the peer-to-peer overlay to find peers. Due to the large differences, it appears that it would be valuable to understand the cause as improving the efficiency of the overlay search would likely result in improved offloading from the server.

E. Number of Neighbors in Overlay

One of the findings of [1] was that a surprisingly large fraction (varying between 30%–50% over a week) of peers were disconnected from the overlay (i.e., they have no open TCP connections to any peers). It was conjectured that the mechanisms for peer discovery and overlay maintenance in the Spotify system cause idle users to become disconnected from the overlay. Clients not being connected to the overlay means that available serving capacity is not fully utilized. On the other hand, idle peers becoming disconnected is not necessarily bad, as it reduces a user's incentive to shut down the client when she is not actively listening. Peers not connected to the overlay can still be found via the tracker.

To follow up on this, we investigated how the number of peer connections a client has varies with the amount of time it has been connected. The client reports the number of peers it currently is connected to every 30 minutes from logging in. When collecting data, we ignored the first report sent by a client as these were sent very quickly after login.

Figure 1 shows percentiles for the number of peers a user is connected to as a function of how long they have been logged in. As can be seen, the number of peers a client is connected to in general slowly decreases over time. There is also a clear periodic pattern where the number of peers users are connected to increases at approximately 24 hour intervals (1440 minutes). The periodic pattern could potentially be explained by the individual user's usage pattern being periodic (e.g., listening to music 9–17 every day), or by the fact that the number of users logged in displays a periodic pattern throughout the day.

We also see that bootstrapping a client up to its peak number of peer connections appears to take roughly 1 hour in the Spotify system. Another general trend is that a surprisingly large fraction of peers are indeed not connected to any peers, and this is true over time. This indicates the importance of having a mechanism apart from overlay based search, such as Spotify's tracker, in order to allow such peers to reconnect to the overlay.

V. Conclusion

We have briefly discussed Spotify's peer-assisted protocol and in particular how the peer-to-peer overlay is formed and maintained. We show that by combining a centralized tracker solution with a simple search in the overlay peer discovery can be efficiently implemented. In particular, we demonstrate that small-radius query propagation can be highly efficient in a peer-assisted system with a moderate number of items.

We also provide more general information on network behavior, and find surprisingly small differences in general between the studied network properties when we compare weekdays with weekends. In contrast, we observe significant differences in the performance of peer discovery within the overlay between weekends and weekdays, which cannot be explained by the measured network statistics.

Acknowledgements

We would like to express our gratitude to everyone working at Spotify for their help.

References

[1] G. Kreitz and F. Niemelä, "Spotify – large scale, low latency, P2P music-on-demand streaming," in *Peer-to-Peer Computing*. IEEE, 2010, pp. 1–10.

[2] G. Kreitz, "Aspects of secure and efficient streaming and collaboration," Ph.D. dissertation, KTH Royal Institute of Technology, May 2011.

[3] M. Casado and M. J. Freedman, "Peering through the shroud: The effect of edge opacity on IP-based client identification," in *NSDI*. USENIX, 2007.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, pp. 1672–1687, dec. 2007.

[5] A. Legout, G. Urvoy Keller, and P. Michiardi, "Understanding BitTorrent: An experimental perspective," INRIA, Technical Report, 2005. [Online]. Available: http://hal.inria.fr/inria-00000156/en/

[6] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking (MMCN)*, January 2002.

[7] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 375–388, 2008.

[8] L. Rabbe, "CIO update: Post-mortem on the Skype outage," http://blogs.skype.com/en/2010/12/cio_update.html, 2010, visited 5 July 2011.

[9] D. Stutzbach and R. Rejaie, "Characterization of P2P systems," in *Handbook of Peer-to-Peer Networking*, X. Shen, H. Yu, J. Buford, and M. Akon, Eds. Springer US, 2010, pp. 1253–1276.

[10] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389 (Proposed Standard), Internet Engineering Task Force, Oct. 2008.

[11] S. Perreault and J. Rosenberg, "TCP candidates with interactive connectivity establishment (ICE)," Internet-Draft (work in progress), draft-ietf-mmusic-ice-tcp-08, Oct. 2009. [Online]. Available: http://tools.ietf.org/html/draft-ietf-mmusic-ice-tcp-08

[12] A. Müller, N. S. Evans, C. Grothoff, and S. Kamkar, "Autonomous NAT traversal," in *Peer-to-Peer Computing*. IEEE, 2010, pp. 1–4.