Share with Thy Neighbors

Surendar Chandra (surendar@nd.edu) and Xuwen Yu (xyu2@nd.edu)

Department of Computer Science and Engineering, University of Notre Dame
Notre Dame, IN 46556, USA

ABSTRACT

Peer to peer (P2P) systems are traditionally designed to scale to a large number of nodes. However, we focus on scenarios where the sharing is effected only among neighbors. Localized sharing is particularly attractive in scenarios where wide area network connectivity is undesirable, expensive or unavailable. On the other hand, local neighbors may not offer the wide variety of objects possible in a much larger system. The goal of this paper is to investigate a P2P system that shares contents with its neighbors. We analyze the sharing behavior of Apple iTunes users in an University setting. iTunes restricts the sharing of audio and video objects to peers within the same LAN sub-network. We show that users are already making a significant amount of content available for local sharing. We show that these systems are not appropriate for applications that require access to a specific object. We argue that mechanisms that allow the user to specify classes of interesting objects are better suited for these systems. Mechanisms such as bloom filters can allow each peer to summarize the contents available in the neighborhood, reducing network search overhead. This research can form the basis for future storage systems that utilize the shared storage available in neighbors and build a probabilistic storage for local consumption.

Keywords: multimedia sharing, localized sharing

1. INTRODUCTION

Technologies to create and consume multimedia objects are becoming commoditized and ubiquitous. Centralized video storage solutions such as YouTube and Google video are increasingly popular. Peer to peer (P2P) technologies are also being developed to distribute contents in a scalable fashion. P2P systems develop overlays that are well connected in order to access all the contents available in the system in spite of failing node and network conditions.

However, this research is motivated by application scenarios where multimedia content is shared solely with neighboring nodes. Localized contents can be accessible during network partitions. Localized sharing might be required as part of the object creator's digital rights management (DRM) requirements. Localized sharing is also attractive in scenarios where the network cost for accessing wide area contents is high. For example, users may share their multimedia objects with others on a long haul flight where it is relatively cheaper to use high speed wireless networks within the aircraft while slow and expensive between the aircraft and the rest of the world. Similarly, home users can directly share objects with their neighbors using high speed wireless networks (in scenarios where the houses are in wireless range of each other; common in many urban and sub-urban areas) rather than through their slower broadband connections. Universities and office campuses can enable sharing of contents within their high speed organizational LANs. For example, Vanderbilt University[1] saved $75,000 a year in network costs by locally hosting music services for students to download songs. Microsoft is also touting localized sharing as an important feature in its upcoming Zune media player. The emergence of large local storage on mobile devices coupled with high speed wireless network technologies such as IEEE 802.11 and 802.15 make these sharing scenarios increasingly viable.

Unlike wide area sharing scenarios, localized sharing introduces additional challenges. Large scale centralized systems aggregate a large volume of multimedia contents (online stores such as Apple, Napster and

Rhapsody host millions of objects). Similarly, an user is highly likely to find more contents of interest among a P2P population of millions in Gnutella networks. Without access to interesting contents and good mechanisms to find and access available content, localized sharing might not be deemed useful by the users. Without active user interest, localized sharing would succumb to disuse. On the other hand, localized sharing has the potential to enable cooperation among a small, like-minded set of clients that might organize objects in a consistent fashion, even though it is highly unlikely to have the variety of content provided by YouTube. Localized sharing has the potential to enable richer forms of sharing semantics; semantics that may not scale to a full P2P deployment.

Rather than hypothesizing the behavior of localized sharing, our goal is to analyze the actual behavior of a system that shares contents exclusively with local neighbors. One instance of an application that enables such sharing is the Apple iTunes software.[2] Apple iPod has emerged as the dominant mobile multimedia device. Recent studies by Student Monitor[3] have highlighted the immense popularity of iPods among undergraduate students; 42% of students own an iPod.[1] iTunes is primarily used to synchronize the multimedia contents between the iPod[4] and a computer; Apple makes this software available for the Microsoft Windows and Apple OSX platforms. iTunes also uses Zero Conf technologies[5] such as DNS service discovery and digital audio access protocol (DAAP)[6] to share and locate objects within the same sub-network. iTunes allows the users to store audio, video and PDF objects (though PDF objects cannot be shared). iTunes imposes relatively few onerous DRM restrictions for such sharing. If the popularity of iTunes (because of iPod) resulted in users also sharing contents, then this environment would be a good target for studies on the nature and behavior of localized sharing of multimedia contents. This paper describes such a study.

We analyze the current sharing behavior of iTunes users in a University setting. All the users in our study were associated with the university, though we do not know their specific categorization: they could be undergraduate, graduate, faculty, staff, computer science majors, liberal arts majors etc. The primary research goals of this paper is to answer the question: *How likely are we to find contents of interest among a small knit community of users?*. This question is similar to *How likely are we to find a neighboring passenger in a long haul flight who is interested in holding a worthwhile conversation?*; we need neighbors who are interested in talking with us and have similar interests. Answer to this question depends on the individual application semantics, the set of neighbors and our notions of *interesting content*.

Our experimental results show that users are already actively making available a diverse range of objects. In our traces on two sub-networks within the university, we noticed that users shared as much as 2.454 TB of objects. Our analysis showed that the current iTunes model of users explicitly connecting to each share and browsing for content could lead to poor access behavior for the users. We show that a simple mechanism that can send a digest of each client contents can allow these sharing systems to generate distributed playlists in a fairly powerful fashion. Our system can effectively allow users to browse and listen to a wide variety of content. Kirovski et. al.[7] are addressing the economics of allowing users to distribute (and sell) contents to immediate neighbors.

The rest of the paper is organized as follows: Section 2 describes our experiment setup. We describe our research results in Section 3. Section 4 places our work in the context of prior research. We conclude with a general discussion of our target application scenarios in Section 5.

2. SYSTEM ARCHITECTURE

In this section, we briefly describe our objectives, iTunes system architecture and the experimental setup. We will highlight our experimental results in the next section.

2.1. Objectives

Our experiments are designed to answer the following questions:

- *Is there anybody out there?* Our work depends on the availability of users who are already sharing significant number and types of objects. If iTunes sharing was not significant, then the rest of the system cannot be developed.
- *What are the appropriate sharing models and semantics?* Localized sharing will not provide the variety of contents of a larger scale system. We seek to understand whether this requires a rethinking from the traditional P2P systems.

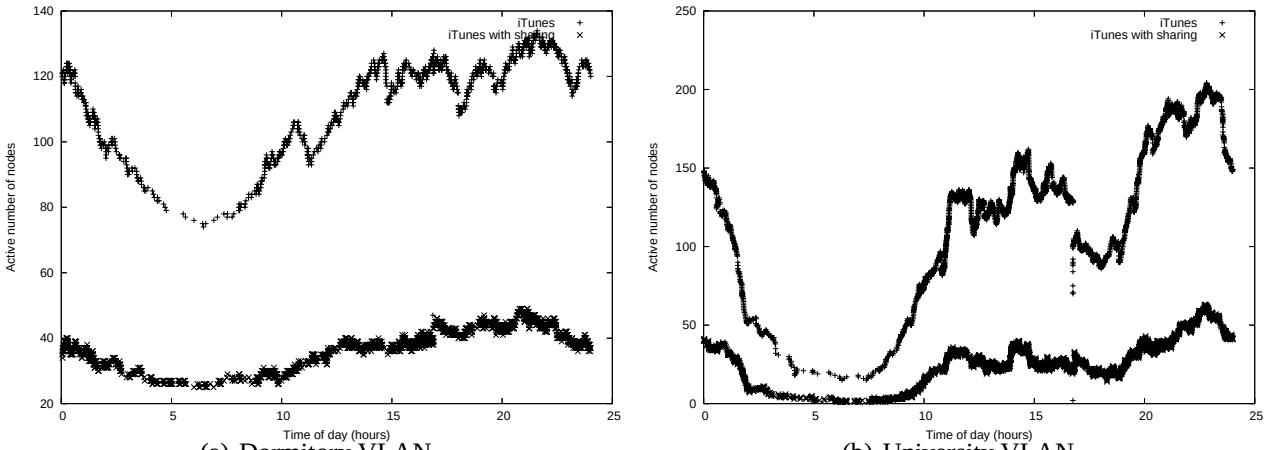
2.2. Experiment setup

For our experiments, we analyzed the songs that were shared using Apple iTunes in an University setting. iTunes allows users to selectively share (using playlists) their song collection with other iTunes clients; this sharing is turned OFF by default. Once the sharing is turned ON, the default behavior is for the clients to share all their local objects. The default iTunes behavior is to free-ride and seek shared collections from other users (this behavior may also be turned OFF explicitly). Recent versions of iTunes restricts the sharing to be within the same sub-network. Also, iTunes only allows sharing objects with five distinct IP addresses within a 24 hour period (even for contents such as podcasts that may allow unrestricted distribution). Users can password protect their shares in order to avoid this five user limitation. Songs purchased from the iTunes store are protected by Apple Fairplay DRM. The computer that can play these songs has to be explicitly authorized and the objects cannot be publicly consumed (though the songs themselves can be shared). iTunes allows users to manage PDF, audio and video objects, though PDF objects cannot be shared with other clients. We collect information about these shared-protected contents in our experiments. Note that our work is not motivated to enable illegal sharing of existing songs. We are interested in the potential for sharing a richer set of legal objects.

AppleRecords[8] is a Java daap client. We modified AppleRecords to connect to iTunes clients and log the songs that were shared by the user. For each of the shared objects, we logged the following attributes: track id (set by the particular iTunes client), track name, album name, artist name, track number, genre, user song rating, object format (e.g. MP3, AAC audio, MPEG-4 audio book), length of object (in milliseconds), sample rate (e.g. 44100 Hz), song bitrate (in kbps), object size (in bytes), BPM, disk count, disk number, song description, comments, date song was added and the date that the song was last modified. We conducted the experiment during late April/early May in 2006. Since Apple restricts sharing within the same sub-network, our collection agent had to be run within the sub-network. Notre Dame partitions its local network in a number of different VLANs. Some of the VLANs include campus wireless, VLANs based on the department affiliation and multiple VLANs inside the dormitory. We monitored one of these dormitory VLANs as well as several wired and wireless VLANs within the campus. We also collected some preliminary traces from another school (which wished to remain anonymous). We did not notice any significant content differences between the users from various networks.

3. RESULTS

In this section, we describe the results of our analysis of the iTunes shared contents logs. First, we analyze statistics about the kinds of objects that are shared by users. Next, we analyze the appropriate sharing models. We use these to develop techniques that can effectively identify whether potential users will find a particular sharing fruitful.



(a) Dormitory VLAN (b) University VLAN

Figure 1. Number of shared iTunes clients by the time of the day

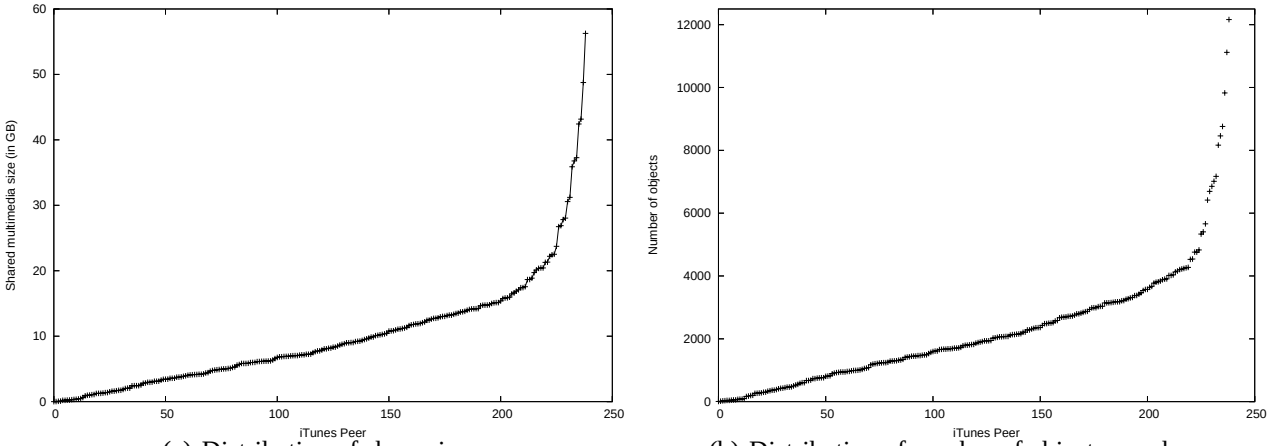
3.1. Is there anybody out there?

The Student Monitor study[3] found that undergraduate students consider the ipod device as *in*; we wanted to know whether this popularity translated to the sharing primitives of iTunes.

First we monitored our dormitory and campus wired VLANs using the Zero conf service discovery capabilities of *dns-sd* tool.[9] On startup and exit, every iTunes client sends out a DACP message, even if the client was not sharing (or seeking shared) contents. iTunes clients that share contents also send a DAAP message. We monitored these requests for a week (using the commands 'dns-sd -B _dacp._tcp' and 'dns-sd -B _daap._tcp', respectively) and logged 1435 unique iTunes clients (measured using DACP) of which 468 clients were sharing contents (measured using DAAP). We consider these numbers to be significant. As a medium sized University, Notre Dame's population includes 1700 graduate students, 6357 undergraduate students and about 1586 faculty and administrators. Assuming that every user was associated with a single machine, almost 15% of the University population used iTunes on the monitored VLANs.

We plot the number of iTunes clients and the number of iTunes clients that were sharing objects by the time of the day in Figure 1. We plot the graph as follows: if a new iTunes client was detected (via monitoring DAAP), we incremented the count of iTunes that are currently online. When the client left, we decremented the count of clients. We noticed that the shapes of these graphs were similar across different days in our traces. Both the dormitory and university population exhibit a bimodal distribution. The iTunes clients within the University network show lower activity during 1 AM to about 10 AM. On the other hand, the clients in the dormitory show reduced activity from around 4:00 am to 10 AM. The University extends the dormitory VLAN into the student center, a popular location for students to work during the daytime. University VLAN is fairly devoid of shared clients during late nights. In general, it appears that many of our users do use iTunes even though sharing from neighbors is not an viable option in early morning. Note that these logs were collected from a smaller subset of the VLANs than the traces collected in the next step.

Next we connected to the individual iTunes shares (clients that have sharing ON and would respond for DAAP queries). Our traces encountered 620 unique iTunes shares (these traces were collected over a larger set of sub-networks than was collected for the previous experiment). Of these, 145 shares were password protected (we do not attempt to break the passwords), 313 shares were busy (client returned an HTTP busy error code; iTunes is a mini-HTTP server) and 111 shares were being blocked by the (default) network firewalls. Firewalls that protect against iTunes requests would block our attempts to connect to the iTunes share and our logger



(a) Distribution of share size (b) Distribution of number of objects per share

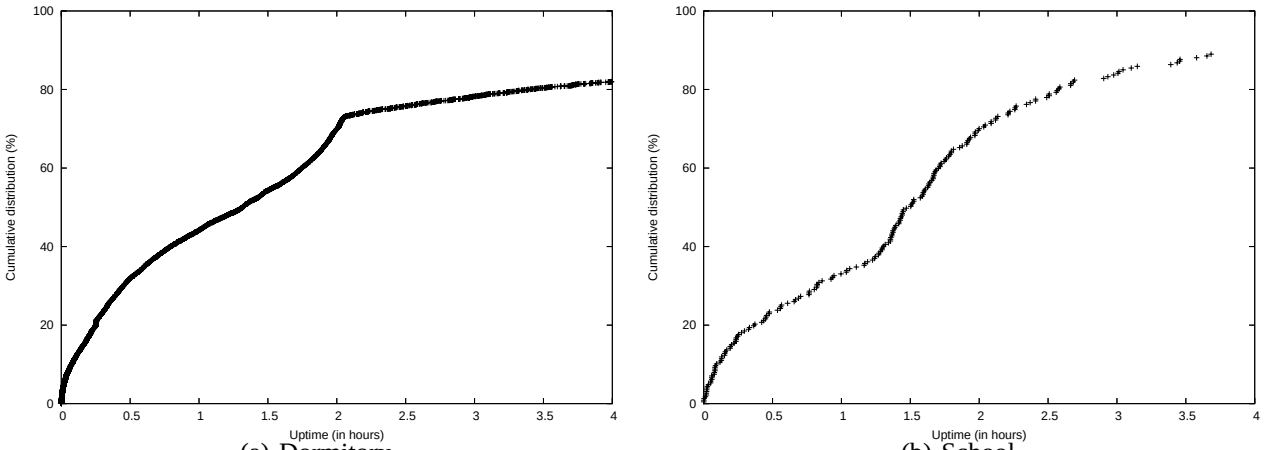
Figure 2. Sharing characteristics of iTunes clients

will timeout. It is possible that there is race condition and that some of these users timed out because the user went offline between the time when we saw them and when we attempted to connect to the share. Clients were likely busy because the tracing program violated the 5 unique IPs addresses per 24 hour restriction. Sometimes it appeared as if the client was just genuinely busy. We successfully collected the share logs from 239 unique users. These users shared 533,768 objects of which 171,068 objects were unique. These systems shared 2454.58 GB worth of data, of which 858.11 GB of data was unique. The rest of our analysis was performed on these traces.

Next, we plot the size of the shared space and the number of objects in Figure 2. We note that about 150 of the 239 clients shared between 5 GB and 20 GB each. Similarly most of the clients shared between 1000 and 4000 objects. These point to significant opportunities for sharing and finding interesting content.

Next we analyze the kinds of objects shared in the iTunes network. Our traces show that of the 533,768 objects shared in the iTunes network, most objects are stored as MP3 or AAC objects. MP3 audio files form a majority with 298,816 objects while AAC audio files make up for 209,716 objects. 14,051 objects were AAC audio objects that were protected by Apple Fairplay DRM technology (likely purchased from iTunes store). There were 598 protected MPEG-4 videos (likely purchased from iTunes store), 226 Apple MPEG-4 video objects (186 of these were podcast objects), while we noted 279 MPEG-4 video objects. There were 366 Quicktime movies and 96 MPEG Audio books. Other formats include MPEG audio file (82), MOV movies (74), MP2 audio (29) and MPEG Audio streams (5). Essentially, audio files dominate the workload. Incidentally, iTunes itself is still predominantly tuned towards audio objects, defining meta-tags such as bits per minute (BPM) even for PDF documents. Apple only introduced video playback capability with version 4.8 in May 2005 and the ability to purchase videos with version 6.0 in October 2005. Hence, we focus on audio playlists for the rest of the paper. It is highly likely that the AAC audio files were ripped on the iTunes client, though it is harder to guess the provenance of the MP3 objects. These songs could have been ripped on the iTunes client, users might have transcoded an object that they purchased from iTunes store to a MP3 or they were illegally downloaded through another P2P system.

Next we plot the uptime of the various iTunes clients in the dormitory network and university networks as a cumulative distribution in Figure 1. We note that about 80% of the clients were online for about two hours or less with a few clients online for a longer duration. The clients in the dormitories were more stable. This was understandable because the university network included wireless LAN networks where we expect users to



(a) Dormitory (b) School

Figure 3. iTunes uptimes

disconnect often (unlike desktops connected to wired LANs). These results show that there is a potential for useful content sharing (shares are around long enough).

3.1.1. Summary of results

To summarize, we noticed that iTunes is indeed popular in our University, about 15% of the users appear to use iTunes, we captured the shared contents from almost 2.4% of the entire user population. On average, users shared over 2,000 songs, contributing about 10 GB of shared space per user for a total of 2.454 TB of data . The clients were around long enough (over two hours) for interesting sharing possibilities.

3.2. How should objects be shared with neighbors?

In the previous section, we showed that there are a significant number of objects available for sharing in the immediate network vicinity. In this section, we discuss the general sharing models that are appropriate for systems that only share contents with the immediate neighborhood. Note that iTunes only allows for objects to be streamed and not downloaded. Future systems that may operate on a richer set of objects might allow objects to be downloaded and replicated.

For simplicity, we assume that during active periods, about 40 clients are online (the actual data exhibited a bimodal distribution. More detailed data was presented in Figure 3). Based on our observation of 239 unique clients during our trace interval, we assume that an object should be available in at least six different clients in order to be available. Though simple, this gives us some rough information to analyze the traces in this section. Also, for our analysis, we performed some rudimentary cleanup of attributes; e.g., normalized the case and removed white spaces. We use this same cleanup for all our analyses in this section.

3.2.1. Searching for exact song names

First we investigate the viability of searching and finding objects by their names. In general, songs that are available in all the 239 clients are popular and are always available. However, they are also available locally and so are not interesting targets to share. On the other hand, songs that are only available in one node is rare. As discussed in Section 3.2, we prefer objects that are available in six or more clients.

We plot the popularity distribution of the songs in all the 239 iTunes clients in Figure 4. Of the 152,850 unique songs analyzed, 138,113 songs (90%) were available in less than seven clients. Non-specific song names

such as *TRACK* $< number >$ and *INTRO* appeared more than once in several clients. Most of the songs are unique implying that the iTunes users are providing a rich trove of music. On the other hand, the availability of these objects are poor; on average only about 40 of the 240 clients analyzed were online at any one time.

In general, systems that strictly share contents with their neighbors would not have access to the rich set of objects that are possible with a more traditional (and scalable) system. Users are unlikely to find specific objects; rather, they are more likely to find objects that belong to a broader category. Users might want objects that are similar to their own collections but actually dissimilar for the specific object. For example, users might browse for songs that belong to the same genre/album/artist while excluding particular objects that they themselves have. With the appropriate categorization, iTunes can potentially provide the users with a wide variety of songs (as many as 90% of the songs are unique). Next, we analyze whether such a categorization exists among iTunes users.

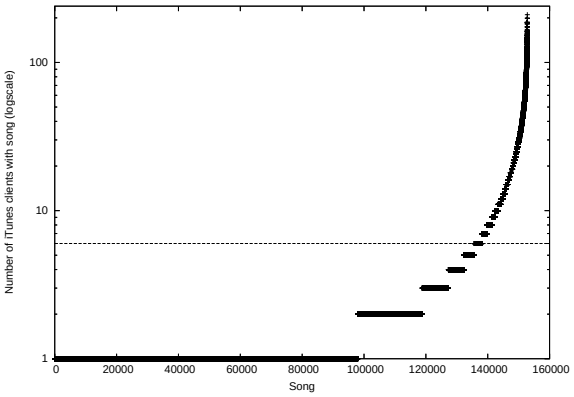
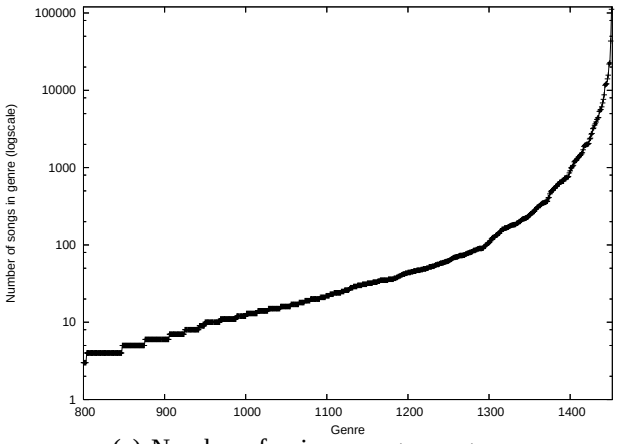


Figure 4. Number of iTunes clients with song

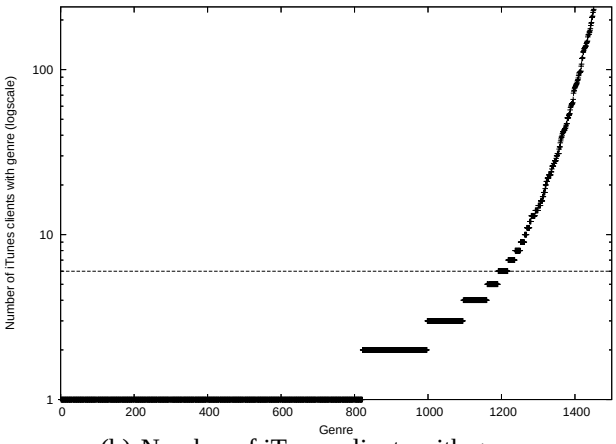
3.2.2. Clustering songs by artist, album or genre

Next, we analyze the songs in order to understand whether it made sense to cluster the songs into some song categories. For this study, we categorize songs in the same genre, album and songs by the same artist. We plot the popularity distribution of the number of clients that have objects from the same category in Figures 5(b), 5(d) and 5(f), respectively. Basing on the argument advanced earlier (Section 3.2), we require that the categories be present in at least six clients in order to be available to an user. Unlike in the case of song names, the availability of the songs in these categories in all the clients are preferable. We also plot the number of objects that belong to each member of the category (songs in the same genre, songs in the same album and by the same artist) in Figures 5(a), 5(c) and 5(e) respectively. Ideally, we would like the number of songs available in categories such as albums, artists and genres to be high.

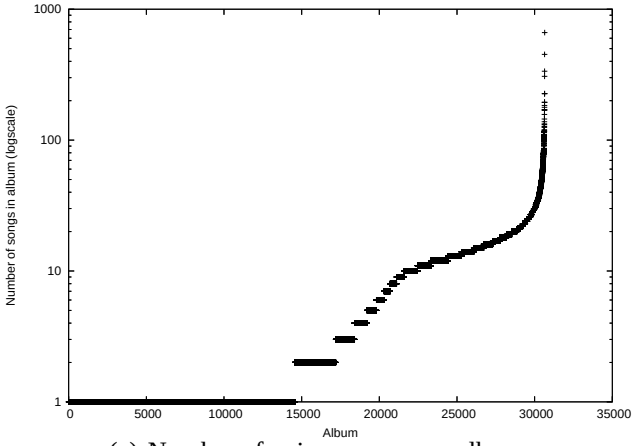
First we analyze the behavior for grouping the available objects based on the genre. By default, iTunes is shipped with 24 genres (including *Unclassifiable*) though users are free to change the genre name. For the most part, we expected that users would stick to this categorization. However, on analyzing Figure 5(b), we were surprised that the users had a wider variety of genre categories; 1,452 genres in total. Also, 99,987 songs (18.7%) did not have any genre. Many of these genre were variations of existing genres (e.g., *ROCK OPERA*). Of these, 1219 genres (about 84%) occurred in six or less clients, potentially affecting their availability. Figure 5(a) shows that many of these genres have tens of songs. Essentially, individual users create genres and populate them with tens of songs. The genres which accounted for more than 1% of the songs are: 1.01365% - *hip-hop*, 1.02425% - *indie*, 1.22965% - *blues*, 1.28383% - *classic rock*, 1.30043% - *metal*, 1.40947% - *jazz*, 1.58444% - *r&b*, 1.76472% - *rap*, 2.01738% - *classical*, 2.67854% - *hip hop/rap*, 2.76453% - *alternative*, 2.79657% - *rock/pop*, 3.24058% - *other*, 3.62095% - *country*, 5.03964% - *pop*, 5.22591% - *soundtrack*, 10.0168% - *alternative & punk* and 25.7948% - *rock*. We conclude that genre might not be a good way to categorize contents because genres appear to be a personal choice. 18.7% of the songs have no genres and of the remaining genres, some of them appear to mean the same category. For example, the genres *hip-hop*, *rap*, *hip hop/rap* might refer to the same category. Many of the categorizations are individual to the users, however they might not realize it because from their perspective, the number of local songs in a certain genre creates the illusion that the particular categorization is popular on other clients as well unless they search all other clients.



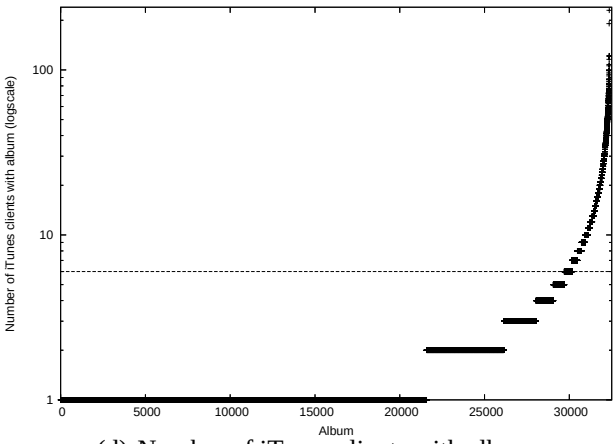
(a) Number of unique songs per genre



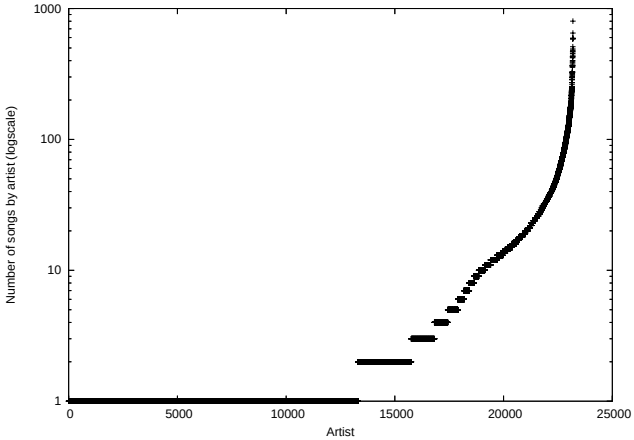
(b) Number of iTunes clients with genre



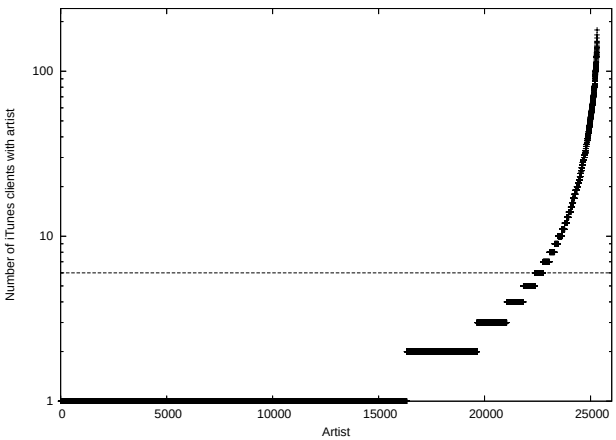
(c) Number of unique songs per album



(d) Number of iTunes clients with album



(e) Number of unique songs per artist



(f) Number of iTunes clients with artist

Figure 5. Popularity distribution of albums, artists and genre

Next we analyze the songs based on their albums. Unlike genres, album names are decided by the artist and one expects them to be unique. Our traces had songs from 32,353 unique albums, though 30,159 albums (93%) occurred in less than seven clients while 43,540 (about 8.1%) songs did not have an album name. Also, an analysis of Figure 5(c) shows that 14,568 (about 45%) of the albums had only one song while 9,055 (28%) albums had ten or more songs. The most popular album (occurring in 191 clients) is *GREATEST HITS*, a generic album title that was created by multiple and unrelated artists. The next most popular albums were *Songs About Jane* by *Maroon 5* and *A Rush of Blood to the Head* by *Cold play* which appears in about 122 clients. Typically, one expects audio albums to consist of about ten songs. Suppose a particular client has two songs from an album, it can expect to find eight more songs to complete the collection. Also, albums with one song in the entire system are not worth searching for in other clients (because that one song must have been from the requestor client). Songs for which the requestor had ten or more songs are probably not worth searching because the requestor might already have all the songs available in that album. We note that 73% of the albums in our traces had either one song or more than tens songs locally.

Next, we analyze the songs based on their artists. Note that a song can be performed by many artists. Our traces showed songs from 25,309 unique artists though 22,674 (89.5%) artists were associated with six or less clients. From Figure 5(e), we note that 13,316 (52.6%) artists had only one song in our system, clients that host these artists and searching the system to find more choices will fail because all the songs that are available in the system are already available in the local system.

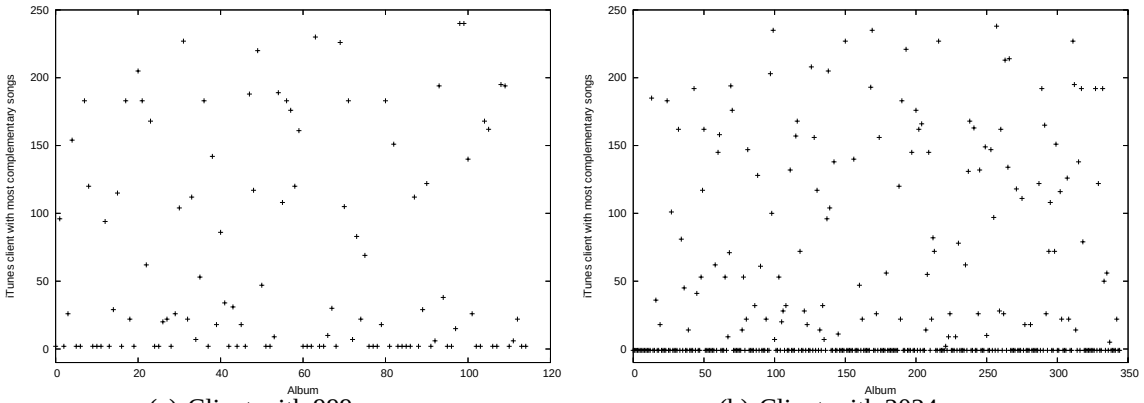
Summary of results We believe that categorizing songs is necessary in order to find complementary songs in systems that restrict the sharing to local nodes. However, many of the categories are unique. An object access mechanism that bases related objects on categories that it possess should avoid searching for categories that are not popular. Seemingly, about 90% of these categories will not lead to related objects as the only client that hosts these objects are themselves. Searching on artist names appears promising as only 52.6% of artists had only one song in the system, the rest has many more songs.

3.2.3. Are iTunes like sharing models appropriate?

Next, we investigate whether the sharing model supported by current iTunes client would lead to the users finding other objects. iTunes users are expected to attach to a new share (providing the password if the share was password protected). Once attached, the users can browse for interesting songs until they voluntarily disconnect. The identity of the sharing users are unknown to the provider. The only way to control sharing is to either turn OFF sharing or by exiting the iTunes client. The sharing is sequential; the user cannot mix songs from two different clients to create a more complex play-list.

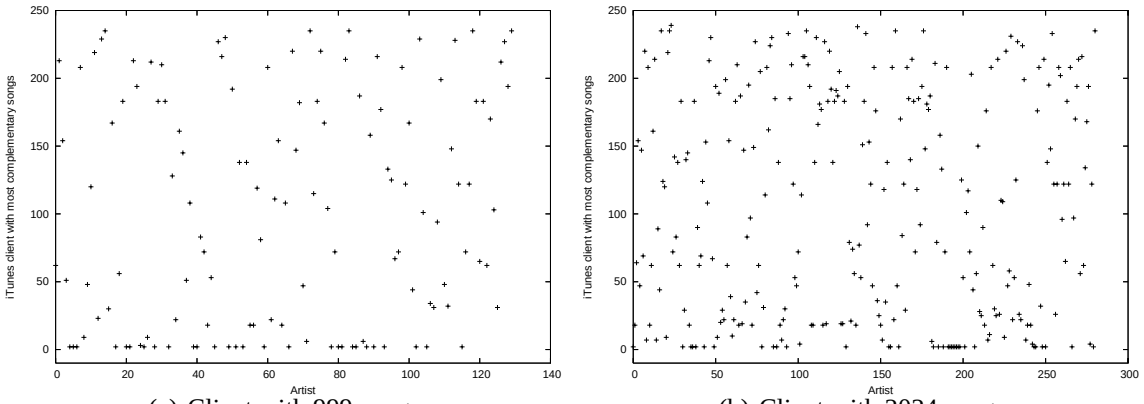
In general, objects of interest for a particular user depends on various factors. Factors such as the user's current mood (e.g., *slow jams* for a mellow evening) cannot be easily measured automatically. On the other hand, the contents of their own collection might give us hints on selecting more interesting contents. For example, one can assume that users would like to hear the remaining songs from albums that they already own or listen to new songs form artists that they may already own. We do acknowledge the limitations in these assertions. For example, users might have carefully hand picked the songs from an album and hence do not want to listen to any more songs from the album. Regardless, it may be possible to automatically create useful playlists that map users own song collection and identify a set of potential clients that provide complementary songs that the user would find interesting.

The first challenge is to identify whether the current iTunes sharing model can allow such mechanisms that create dynamic playlists. Specifically, we want to investigate whether connecting to a single share can provide a rich set of complementary songs based on local albums and local artists. For these experiments, we analyzed



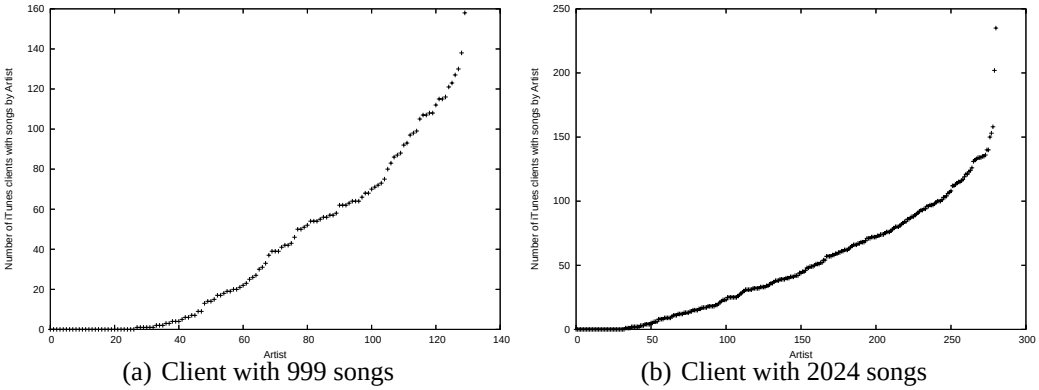
(a) Client with 999 songs (b) Client with 2024 songs

Figure 6. Best client to host complementary albums



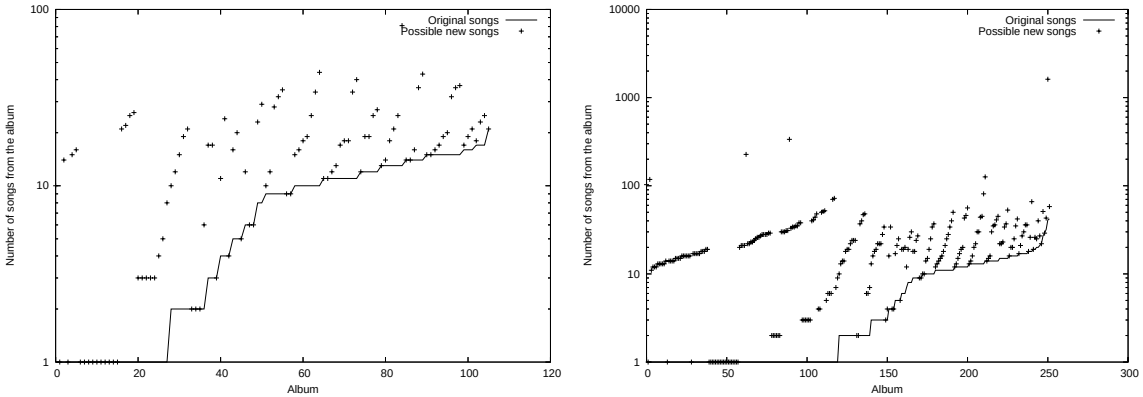
(a) Client with 999 songs (b) Client with 2024 songs

Figure 7. Best client to host complementary artists



(a) Client with 999 songs (b) Client with 2024 songs

Figure 8. Clients that host complementary songs by artist



(a) Client with 999 songs (b) Client with 2024 songs

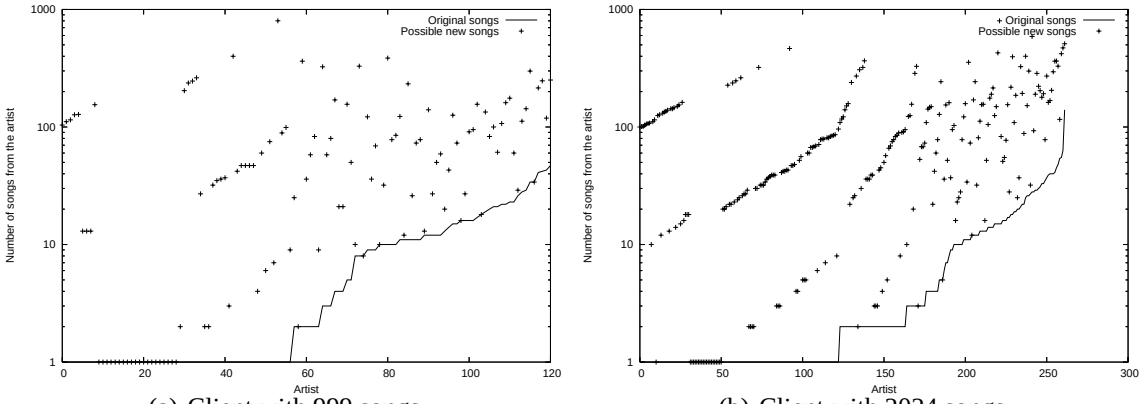
Figure 9. Count of number of songs per album in local collection

this question for each individual client. Specifically, for each client, we identify all the albums (or artists) that the user owns and for each of these albums (or artists) we find the client that provides the most complementary songs. We define this client by counting the number of songs that the remote client has in this particular category (same album or song) that are not present in the local client. We then plot a graph between the local album (or artist) and the best client that provide complementary songs for two representative clients that hosted 999 songs and 2024 songs in Figures 6 and 7, respectively. Ideally, we would like a graph that lists few shares as potential remote clients. Fewer data points show that the user only needs to connect to a few of these *good* clients. However, we notice that the peers with the most complementary songs is truly random; with different clients providing a good collection of objects for different albums (or artists). Hence, we believe that the iTunes sharing model would require too many explicit connection switching from client to client. Currently, users are required to manually change the shared hosts with no mechanisms to give hints on choosing good peers. Without such help, the user is likely to stay with clients which may provide good collection for one album (or artist) but not so good for another album. Note that this graph assumes that the user might choose to listen to a particular album (or artist) in random and that all the remote clients are available online all the time. Users who only listen to a particular album as well as scenarios where many of the remote nodes are offline might see different behavior. Also, the user who only listens to a particular artist may find the best complementary node by chance.

Now, not all nodes that were logged by our system was online at the same time. Hence, we analyzed whether the system that hosts complementary songs are robust. For each artist picked on the client, we counted all the peers that hosted some complementary songs and plot the results in Figure 8. Higher values signify that there are a number of other nodes that host complementary songs. We note that popularity distribution depends on the particular artist; not all artists have many clients with some complementary songs.

3.2.4. What are the appropriate access models?

So far, we have shown that iTunes users share a substantial amount of data. However, the current iTunes sharing model of randomly connecting to a client and expecting to find interesting set of related objects is not appropriate. A better search mechanism for access in this environment might be to group objects by categories such as genre, album and artist. Individual clients would then choose objects from each category and look for songs that are complementary to their collection. Since many of the songs/albums/artists are unique, the challenge is to see whether there is a correlation between categories that are popular locally and whether that will guide the search. Otherwise, users might select a local category (album, artist etc.) and fruitlessly search for complementary contents online. Note that about 90% of the album and artist names were associated with a single object and



(a) Client with 999 songs (b) Client with 2024 songs

Figure 10. Count of number of songs per artist in local collection

hence a node that has the album and searching for new content will not locate any more contents. For our analysis, we focus on the categories of songs from the same album and same artist. For the two typical clients investigated in the previous section, we analyze the number of songs per album or artist in the particular clients collection and plot them in Figures 9 and 10, respectively. We also plot the total number of songs for each of the categories available in our entire system (available in all the 239 clients). If the exclusive provider for this particular category was the local client, then both these measures would match. We prefer higher number of objects in other clients. Note that, on average only 40 clients were online at any one time. The actual results would be far sparser. Also, for the search, we prefer a graph that shows a correlation between songs in the local collection and complementary songs in other clients. A correlation would mean that one can analyze the local song distribution and predict the album (or artists) that should be explored for remote access. However, from the plots, we note that there are no trends between the song collection in the local node and the global system. For example, albums (or artists) with just one song locally are equally likely to host many songs remotely or none at all. Without such a correlation, we require other mechanisms that will allow the local node to know whether a particular artist/album is worth searching for in the immediate neighborhood.

One way to accomplish this goal is a reactive policy that sends a bloom filter[10] representation of the local songs (within the category) to the remote clients and have them report back songs that were not found to be represented in the bloom filters. Such a mechanism would require one request to be sent to all the online clients, each of whom will check whether they have any objects of the same category. Clients which host matching category objects will check their local collection for complementary songs by trying to match all local songs (of the given category) with the bloom filter representation. Bloom filters exhibit no false negatives and so all the complementary songs are identified. The client can then send this list of complementary songs to the originator, potentially requiring a response from all the clients that are online. However, as we noted in Section 3.2.2, a significant number of songs from categories such as same albums and artists are not replicated and are unique to a single host. Hence, many of the requests for complementary songs will go unanswered, wasting a RTT timeout for the requestor.

Another possible mechanism is a pro-active approach wherein the remote nodes push a bloom filter representing all the songs in the local client and a bloomier filter[11] to give the count of objects within the category to all the neighbors. The local node can then figure out how many complementary songs each clients hosts (by counting the number of local songs that match the bloom filter and then computing the difference between the bloomier filters and the local count) and then send a targeted query to the specific node. In this approach, the original representations can be piggy backed once, perhaps with the DACP mechanism. The local client can

perform the matching operations locally and only send a targeted request to the client that hosts complementary songs. Compared to the previous approach, this approach will lead to fewer network messages because many categories are unique and should not be searched over the network (Section 3.2.2).

For the above search mechanisms, bloom filters can compactly represent the directory information with no false negatives and a low false positive rate. Ideally, we prefer the representation to fit into a single network packet (typically 1500 bytes in the local LAN). Hence, we analyzed[12] the number of hash functions required for a given bloom filter (computational overhead) and the false positive rate; ideally we require small false positive rates. For a 1500 byte packet with 1000 objects, the optimal number of functions is 8 and a false positive rate of 0.3%. However, for 10000 entries, the number of functions is one with a rate of 56.5%. In fact, we require 15000 bytes (over 10 packets) to reduce the false positive rate to 0.3%. In our study, users are likely to host about 10,000 songs and about 1,000 albums (or artists). For representing song names, we need mechanisms such as compressed bloom filters[13] in order to reduce the network packet sizes. An actual implementation which analyzes the tradeoff between these two approaches are the subject of our ongoing implementation.

4. RELATED WORK

Recently, P2P systems are being developed to solve a number of research challenges in large scale data distribution and management. Unstructured P2P systems such as Gnutella[14] organize topologies using local information. Structured overlays such as Chord[15] build a distributed hash table to store and locate contents. Saroiu et. al.[16,17] analyzed the behavior of Gnutella. Zhao et al.[18] analyzed the contents of Gnutella networks. Gnutella does not maintain the rich meta-data information available and provided by iTunes. Rich meta data like what is available to us might also help build better search mechanisms for general P2P systems. Also, P2P systems are designed to scale to a large number of nodes. Unstructured P2P networks suffer from lack of overlay resiliency to node failures. Ripenau[19] and Saroiu[20] argued that although Gnutella is resilient to random node failures, the overlay is susceptible to the failure of the few highly connected nodes. Our work focuses on scenarios where nodes are closer and fewer, allowing for system choices not always possible in large scale systems. Farsite[21] is the closest project to this research in that both operated on storage components that were distributed within a single enterprise. Farsite used tradeoffs such as centralized directories that may not scale to the size of systems that typical P2P systems such as Oceanstore[22] were designed for.

There is a danger that P2P systems, including the system described in this paper will lead to illegal object sharing. RIAA[23] no longer considers illegal song-sharing an uncontrolled threat within the USA. We also made sure that we do not break any DRM restrictions already implemented by Apple. Even though in our study we have no way of ascertaining the song provenance, the iTunes sharing hasn't been challenged by RIAA within the USA.

5. DISCUSSION AND FUTURE WORK

The motivation of this work is not to enable newer mechanisms to illegally share songs; rather, we investigate mechanisms that relies on a small, local population to provide interesting objects for sharing. We analyzed the behavior of Apple iTunes as it is already widely deployed. We note that there are significant number of unique objects available in this systems. The challenge is to cluster them and develop mechanisms that will allow the local peer to make decisions on the objects to search, thus avoiding unnecessary network traffic. Our ultimate hope is to leverage systems such as iTunes to provide a richer (with probabilistic availability) storage abstractions.

ACKNOWLEDGMENTS

This work is supported in part by the U.S. National Science Foundation through grants IIS-0515674 and CNS-0447671.

REFERENCES

1. N. Timiraos, "Free, legal and ignored." Wall Street Journal, July 6, 2006 - Page B1, July 2006.
2. "Apple itunes software." http://en.wikipedia.org/wiki/Itunes.
3. S. M. LLC, "Lifestyle and media, spring 2006." http://www.studentmonitor.com/.
4. "ipod." http://en.wikipedia.org/wiki/Ipod.
5. "Zero configuration networking (zeroconf)." http://www.zeroconf.org/.
6. "Digital audio access protocol (daap)." http://daap.sourceforge.net/.
7. D. Kirovski and K. Jain, "Off-line economies for digital media," in *ACM Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '06)*, pp. 118–123, (New port, RI), May 2006.
8. C. Davies, "Applerecords." www.cdavies.org/applerecords.html.
9. "Dns service discovery." http://www.dns-sd.org/.
10. A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics* **1**(4), pp. 485–509, 2004.
11. B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The bloomier filter: an efficient data structure for static support lookup tables," in *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 30–39, Society for Industrial and Applied Mathematics, (Philadelphia, PA, USA), 2004.
12. P. Manolios, "Bloom filter calculator." http://www-static.cc.gatech.edu/~manolios/bloom-filters/calculator.html.
13. M. Mitzenmacher, "Compressed bloom filters," in *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pp. 144–150, ACM Press, (New York, NY, USA), 2001.
14. J. Frankel and T. Pepper, "Gnutella." http://gnutella.wego.com/.
15. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 149–160, ACM Press, Aug. 2001.
16. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking 2002*, Jan. 2002.
17. K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pp. 314–329, ACM Press, 2003.
18. S. Zhao, D. Stutzbach, and R. Rejaie, "Characterizing files in the modern gnutella network: A measurement study," in *Proceedings of SPIE/ACM Multimedia Computing and Networking*, **6071**, (San Jose, CA), Jan. 2006.
19. M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal* **6**(1), 2002.
20. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, (San Jose, CA, USA), January 2002.
21. A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer, "Farsite: federated, available, and reliable storage for an incompletely trusted environment," *SIGOPS Oper. Syst. Rev.* **36**(SI), pp. 1–14, 2002.
22. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "Oceanstore: an architecture for global-scale persistent storage," in *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pp. 190–201, ACM Press, 2000.
23. J. Graham, "Riaa chief says illegal song-sharing 'contained'." USA TODAY, June 2006.