An empirical analysis of serendipitous media sharing among campus-wide wireless users

SURENDAR CHANDRA
Unaffiliated
and
XUWEN YU
VMWare Inc.

Contemporary systems use centralized as well as peer-to-peer mechanisms for the large scale distribution of media objects. In this work, we investigate a serendipitous mechanism for directly sharing media objects among a local community of wireless users. This localized sharing is attractive when wide area network connectivity is undesirable, expensive or unavailable; especially when the shared media objects are large. With some restrictions, such localized sharing of media objects is also acceptable to content owners. However, localized sharing has to contend with far fewer media providers who may also not offer the variety of objects available from wide-area services. We collected empirical data from the widely deployed Apple iTunes application for our analysis. We showed that users are already making a significant amount of media objects available for serendipitous sharing. Our analysis showed that the shared object annotations exhibited a Zipfian long tail distribution. The availability patterns of wireless iTunes users and the object annotations makes serendipitous sharing inappropriate for scenarios that require access to a specific object. Instead, mechanisms that allow the user to specify classes of interesting objects are better suited for such users. Also, given the smaller scale of these systems, serendipitous sharing can benefit from approaches that allow users to disseminate a compact representation of their shared objects. Though the wireless user availability rates was not as high as what was observed in a corporate desktop setting, a large fraction of the users showed high temporal consistency. This allows for high availability with reasonable replication during weekday daytime hours. We answer important questions regarding the viability of a campus-wide media sharing system.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed applications

Additional Key Words and Phrases: local media sharing, peer-to-peer

1. INTRODUCTION

Multimedia capable mobile devices (such as smart phones and Apple iPods) are ubiquitous. By September 2009, Apple had sold over 220 million iPod devices with 17.4 million iPod and iPhone devices sold in the quarter ending in June 2009. A similar explosion is observed in other device categories as well. This device ubiquity presents an invaluable opportunity for many media sharing applications. For example, all incoming freshmen to the University of Missouri School of Journalism [Missouri 2009] are required to own a iPhone/Touch in order to view audio and video course materials.

There are a variety of mechanisms to distribute media objects. Centralized services such as the Apple iTunes Store play an important role in media distribution. The iTunes store [Schiller 2009] hosts over ten million songs and had sold over 8.5 billion songs by September 2009. The media objects themselves are delivered

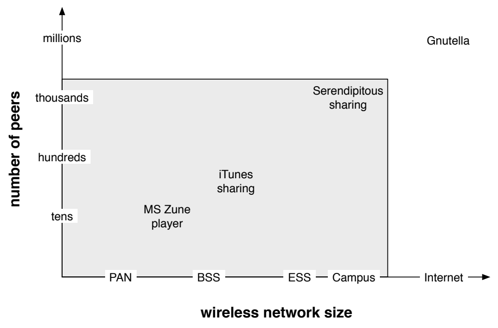


Fig. 1. Variety of peer to peer media distribution mechanisms based on reach

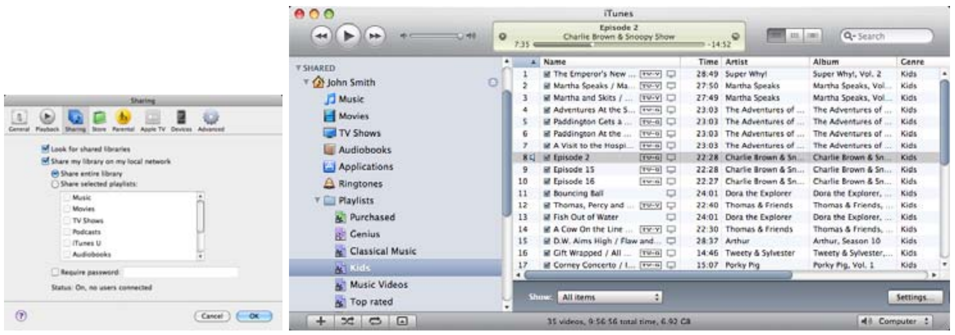
using distributed CDN servers. Recently, there has been some effort to directly distribute the objects from mobile devices through peer to peer (P2P) mechanisms. The emergence of large local storage on mobile devices coupled with high speed wireless network technologies such as IEEE 802.11n and 802.15 makes these sharing scenarios viable. Hence, we focus on wireless distribution mechanisms.

Depending on the reachability range of the mobile devices, different P2P distribution mechanisms had been developed (Figure 1). For example, the Microsoft Zune media player uses WiFi technology to distribute objects to other users who are within a radius of a few hundred feet. The Apple iTunes[1] application can stream objects to users who are inside the same sub-network. Depending on the network setup, a sub-network can reach tens of users within an organization. However, P2P systems such as Gnutella operate on a global scale with millions of users.

Prior efforts had extensively investigated various aspects of global P2P systems; including overlay maintenance [Stoica et al. 2001; Rowstron and Drushel 2001; Chawathe et al. 2003] and query routing [Massoulie et al. 2006]. Systems such as Gnutella and Bittorrent are actively used by millions of users and consume a significant amount of the network bandwidth on the Internet. In this paper, we investigate local media sharing. We differentiate local sharing from Internet scale P2P sharing by referring to it as serendipitous sharing.

Serendipitous sharing offers a number of benefits. Localized sharing can remain accessible during network partitions. It can be inexpensive when network costs for accessing wide area objects are high. For example, it is cheaper to use high speed networks within an aircraft [Lovitt 2008] than it is to use the slow and expensive networks between the aircraft and the rest of the world. Universities and office campuses can enable sharing of objects within their high speed organizational LANs in order to save on Internet access costs. For example, Vanderbilt University [Timiraos 2006] saved $75,000 a year in network costs by locally hosting music services for students to download songs. Localized sharing might also be required as part of the object creator's digital rights management (DRM) requirements. Note that many of these advantages are also shared by approaches that maintain a local replica of a centralized service. On the other hand, serendipitous sharing does not require a priori deployment of the replica server infrastructure.

[1]`http://www.apple.com/itunes/`



[✓] Look for shared libraries
[✓] Share my library on my local network
[✓ Share entire library] Share selected playlists:
[ ] Music
[ ] Movies
[ ] TV Shows
[ ] Podcasts
[ ] iTunes U
[ ] Audiobooks
[ ] Require password:
Status: On, no users connected

Episode 2
Charlie Brown & Snoopy Show
7:35 -14:52

	Name	Time	Artist	Album	Genre
1	The Emperor's New ...	28:49	Super Why!	Super Why!, Vol. 2	Kids
2	Martha Speaks / Ma...	27:50	Martha Speaks	Martha Speaks, Vol...	Kids
3	Martha and Skits / ...	27:49	Martha Speaks	Martha Speaks, Vol...	Kids
4	Adventures At the S...	23:03	The Adventures of ...	The Adventures of ...	Kids
5	Paddington Gets a ...	23:03	The Adventures of ...	The Adventures of ...	Kids
6	Paddington At the ...	23:03	The Adventures of ...	The Adventures of ...	Kids
7	A Visit to the Hospi...	23:03	The Adventures of ...	The Adventures of ...	Kids
8	Episode 2	22:28	Charlie Brown & Sn...	Charlie Brown & Sn...	Kids
9	Episode 15	22:28	Charlie Brown & Sn...	Charlie Brown & Sn...	Kids
10	Episode 16	22:27	Charlie Brown & Sn...	Charlie Brown & Sn...	Kids
11	Bouncing Ball	24:01	Dora the Explorer	Dora the Explorer, ...	Kids
12	Thomas, Percy and ...	22:40	Thomas & Friends	Thomas & Friends, ...	Kids
13	Fish Out of Water	24:01	Dora the Explorer	Dora the Explorer, ...	Kids
14	A Cow On the Line ...	22:30	Thomas & Friends	Thomas & Friends, ...	Kids
15	D.W. Aims High / Flaw and...	28:37	Arthur	Arthur, Season 10	Kids
16	Gift Wrapped / All ...	14:46	Tweety & Sylvester	Tweety & Sylvester,...	Kids
17	Corney Concerto / I...	15:07	Porky Pig	Porky Pig, Vol. 1	Kids

Show: All items

35 videos, 9:56:56 total time, 6.92 GB

(a) Sharing interface (b) iTunes interface for browsing shared contents

Fig. 2. iTunes sharing

In general, understanding the performance of P2P sharing systems requires an analysis of the contents that are shared by users as well as durations when such users are available for sharing. Prior work analyzed the shared objects as well as the queries issued by Gnutella users [Zaharia et al. 2007; Acosta and Chandra 2008b] and the aspects of Gnutella users that affected overlay maintenance [Saroiu et al. 2002; Stutzbach et al. 2007; Acosta and Chandra 2007]. In this work, we perform a similar analysis of a popular serendipitous media sharing system.

1.1 Apple iTunes and serendipitous sharing

For this study, we require a popular serendipitous media sharing application; Apple iTunes is one such application. Apple makes this software available for the MS Windows and Mac OSX platforms. iTunes is primarily used to purchase media objects from the iTunes store as well as synchronize the multimedia contents for an iPod. The iTunes store is the largest music store with over 100 million credit card accounts. The latest version of iTunes supports the Home Sharing feature which allows up to five iTunes clients to synchronize their music library. iTunes also allows users to share their music library with other users. By default, this sharing is turned OFF. iTunes users explicitly attach to a single share (providing the password if the share was password protected) in order to play the shared songs. The client computer that wishes to play songs that are protected by Apple Fairplay DRM needs to be explicitly authorized even though such objects can be shared with no restrictions. Since April 2009, all objects sold in the iTunes store are not copy protected with this restriction anymore. iTunes advertises the sharing status to other clients within the same sub-network using the [illegible] Zeroconf [Guttman 2001] service. The identity of the users who are sharing songs is unknown to the provider. Once the sharing is enabled, the default behavior is to share all the objects (illustrated in Figure 2(a)). Users can selectively share their song collection using play-lists. The default iTunes behavior is to free-ride and seek shared collections from other users (may also be disabled). iTunes only allows sharing session requests with five distinct IP addresses within a 24 hour period (even for contents such as podcasts that may allow unrestricted distribution). Note that a connected user can continue to play the songs even if their IP address changed;

the check is only performed during the initial session establishment phase. Users can password protect their shares in order to avoid this five user limitation. There was evidence that students in our campus were using this feature to circumvent the five user limitation; we observed password protected iTunes shares with names such as [illegible]" and [illegible]. Third party media servers such as firefly[2] also remove this sharing restriction.

The shared songs can be played by other iTunes clients as well as by devices such as AppleTV and Roku SoundBridge. Clients can browse other online iTunes shares; Figure 2(b) illustrates a user browsing a play list called "Kids" shared by "John Smith". iTunes behaves like a HTTP server for servicing requests for shared objects. However, the client component behaves like a streamed system by downloading the shared objects without retaining them for future use. On the other hand, systems such as AppleRecords [Davies] allows the client to download shared objects.

We studied the shared contents by analyzing objects shared by iTunes users in our campus. We analyzed user availability by monitoring the [illegible] and [illegible] services. However, we expand the scope of our analysis to include a more general serendipitous sharing application. Note that some design decisions are driven by the desire of content providers to protect their intellectual property. Thus, the validity of only allowing authorized iTunes client to play a DRM protected song is beyond the scope of our analysis. On the other hand, we question the browsing model used by iTunes. This paper makes the following contributions:

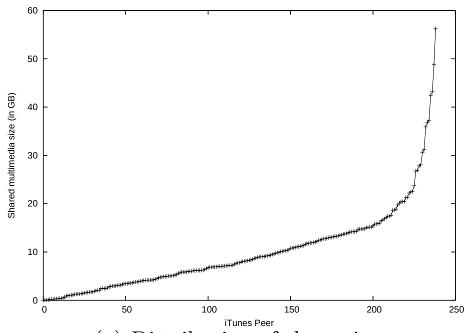
— We experimentally show that users are already actively making available a diverse range of objects. In our analysis of two sub-networks within the university, we observed between 300 and 1,000 simultaneously available iTunes users (depends on the time of the day). Of these users, between 100 and 250 (6%-15% of the total wireless population) iTunes users were making their contents available for sharing by other users. An analysis of shared contents from 239 iTunes users showed that these users shared as much as 2.454 TB of media objects.

— Object annotations such as genre, album and song name exhibited Zipf like long tail distribution where few objects were extremely popular and many objects were rare. The current iTunes model of users explicitly connecting to each share and browsing for songs could lead to poor access behavior for users. In addition, we show that a simple mechanism that sends a bloom filter digest of each client's contents can allow these sharing systems to generate distributed play-lists in a fairly powerful fashion. Our system can effectively allow users to browse and listen to a wide variety of contents.

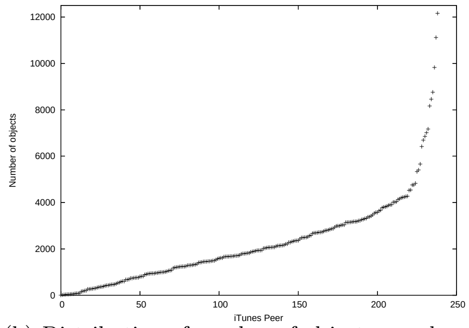
— Using extensive analysis of user behavior, we show that though laptop users were online for shorter durations, their temporal consistency can provide reasonable availability, especially at the times of the day when students were typically active. In addition, reliance on server storage improved object availability.

The rest of the paper is organized as follows:, Section 2 describes the experiment setup and shows that localized sharing is already viable. We describe our research results for analyzing shared contents and object availability in Section 3. Section 4 describes related work with concluding remarks in Section 5.

[2]`http://www.fireflymediaserver.org/`



(a) Distribution of share size



(b) Distribution of number of objects per share

Fig. 3. Sharing characteristics of iTunes clients

2. EXPERIMENT SETUP

Our analysis is based on empirical data of the contents shared by Apple iTunes users as well as on iTunes user availability behavior in our campus. Next, we describe our data collection methodology as well as our evaluation mechanism.

2.1 Empirical data for evaluation

First, we describe the empirical data that is used for our system analysis.

2.1.1 [illegible] For our experiments, we analyzed the songs that were shared using Apple iTunes in an university setting. We modified AppleRecords [Davies], which is a Java daap client, to connect to iTunes clients (clients that had sharing turned ON and that would respond for DAAP queries) and log the songs that were shared by the user. Note that the users did not know that we collected information about the objects that they had publicly shared. 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 the 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 performed some rudimentary cleanup of object attributes; e.g., normalized the case and removed white spaces. We conducted experiments 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 into 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 traces from another school (which wishes to remain anonymous). We did not notice any significant differences between the users from various networks.

Our traces encountered 620 unique iTunes shares. Of these, 145 shares were password protected (we do not attempt to break the passwords), 202 shares were busy (client returned an HTTP busy error code; iTunes is a mini-HTTP server) and

111 shares were being blocked by network firewalls. Any violations of the iTunes' five unique IPs addresses per 24 hour restriction was marked as busy. After multiple attempts, we collected the list of songs from 77 shares which were initially marked as busy. Firewalls that protect against iTunes requests would block our attempts to connect to the iTunes share and our logger would timeout. It is possible that some of these requests timed out because the user went offline between the time when we saw them and when we attempted to connect to their share.

Eventually, we successfully collected the list of shared songs from 239 users. These users shared 533,768 objects of which 171,068 objects were unique. They shared 2454.58 GB worth of data, of which 858.11 GB of data was unique. We plot the distribution of the size of the shared space and the number of objects in each user in Figure 3. We note that about 150 of the 239 clients shared between 5 GB and 20 GB each. Most of these clients also shared between 1,000 and 4,000 objects.

Most objects were stored as MP3 or AAC objects. MP3 audio files made up a majority with 298,816 objects, while AAC audio files made up for 209,716 objects. There were 14,051 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 included 82 MPEG audio files, 74 MOV movies, 29 MP2 audio and five MPEG Audio streams. Essentially, audio files dominated 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 came with version 6.0 in October 2005. Hence, we focus on audio play-lists 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 a) ripped on the iTunes client, b) transcoded an object that was purchased from iTunes Store or c) were obtained illegally.

2.1.2 [illegible] We also collected information regarding when iTunes users were available as well as durations when these iTunes users were sharing their music library. We used the Zeroconf service discovery capabilities of [illegible] tool [dnssd] to monitor the [illegible], and [illegible], service. On startup and exit, every iTunes client sends a DACP message, while also sending a DAAP message during durations when the iTunes was configured by the user to share contents. This service availability information is broadcasted to the monitoring clients using link local multicast. Since these multicast packets are not routed, the monitoring station has to be co-located inside the monitored VLAN. We collected data for eleven days from September 19, 2006 through September 29, 2006. During this time, the entire campus wireless LAN infrastructure was configured to route all Zeroconf service discovery packets to the monitoring station. This allowed us the flexibility of not installing a monitoring station inside each of the campus WLANs. During the two weeks prior to September 25, 2006, about 9,600 computers used our wireless networks: 7,592 running Windows, 1,939 running Mac OS, 72 running Linux and other OSs. Our logs showed 4,893 and 1,702 unique machines using the [illegible], and [illegible], services, respectively. These number were a significant portion of the entire wireless user population.

2.2 Evaluation mechanisms

Next, we describe the design principles of our evaluation mechanism.

2.3 Analysis of objects shared by iTunes users

We analyzed the objects shared by iTunes users in order to understand whether serendipitous sharing will be fruitful. This analysis required a model of user's song interests in order to identify whether objects available from other users will be of interest to them. If no such songs were available then serendipitous sharing will be unfruitful. Such an analysis is typically not performed for Internet scale P2P scenarios because it is assumed that all objects are available somewhere in the system; failure to locate an object is assumed to be an artifact of the overlay maintenance and search mechanisms. However, Acosta et al. [Acosta and Chandra 2008b] showed the fallacy of this assumption among Gnutella users.

In general, objects of interest for a particular user depends on various factors. Factors such as the user's current emotional state (e.g., [illegible] for a mellow evening) cannot be easily measured. However, the contents of their own collection might give us hints on selecting more interesting contents. For example, users may like to hear the remaining songs from albums that they already own or listen to new songs from artists that they may already own. We do acknowledge the limitations in these assertions. For example, users might have carefully selected songs from an album and hence do not want to listen to any more songs from the album. Also, note that our log data does not show whether users were sharing all or part of their music collection. In an article in a college newspaper, Aubrey [Aubrey 2003] described the notion of [illegible] wherein users choose the songs in their playlist to match their social persona. Using the results of interviews with 13 participants, Voida et al. [Voida et al. 2005] observed similar efforts at impression management in a corporate setting. Since users in our campus were observed to share a large number of files, it is likely that users were sharing most of their music library. We assume that users share all their music collection in order to identify songs that they might find interesting and available from other iTunes shares.

Since we assume that users share all their songs, analyzing the shared contents gives us an indication of a particular user's taste. We can then deduce if a user is interested in a particular object as follows: if the user has a copy of the object, they are unlikely to be interested in seeking this object. Thus, if an object is available in all of the 239 users, then the object is too popular and not interesting to share. If all objects were available in all nodes, then serendipitous sharing will be unfruitful. On the other hand, an object that is available in only one node means that the user who has the object is uninterested in searching for it in other nodes. However, every other node is potentially interested in this particular object. Note that we also need to analyze the node availability in order to understand whether the object will be available for other nodes. Another mechanism to seek interesting shared contents is to use the knowledge of the local contents in order to seek objects which are similar and yet complement the local collection. For example, if a user has a song from a particular album/artist/genre, then others songs which also belong to the

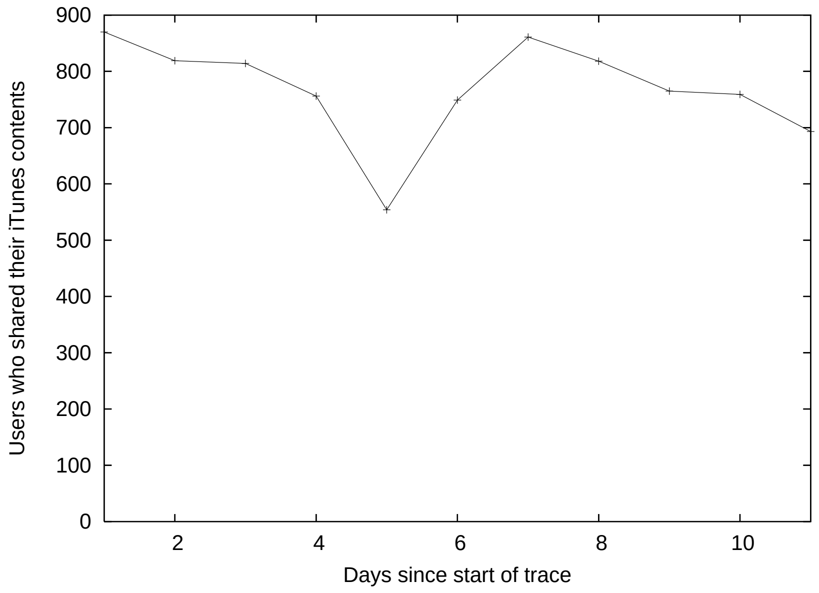


Fig. 4. Number of unique iTunes shares on each day

same album/artist/genre (but not the exact same object) are considered to be of interest.

2.3.1 [illegible] We observed a diurnal pattern with between 100 and 250 users who were simultaneously sharing their contents across the campus (Section 3.2.1). We collected availability information from 1,702 wireless users in our campus (Section 2.1.2). However, not every user was available on each day. We plot the number of unique users seen on each day in Figure 4. We observed about 750 unique users each day. Also, we observed that about 40 clients were online within a subnet. Since iTunes restricts its sharing within a given subnet, the challenge is to reconcile the diurnal user availability patterns in order to draw meaningful conclusions about the availability of shared objects for a serendipitous media sharing system. Increasing the replication ratio will also make it likely for other nodes to have a fruitful sharing experience.

3. ANALYSIS OF SERENDIPITOUS MEDIA SHARING USING EMPIRICAL TRACES

iTunes is popular, about 18% of our wireless users shared their iTunes collection. We captured the shared objects from almost 14% of these users. On the other hand, we fully captured durations when wireless users were using iTunes as well as durations when they were sharing their music collection. Next, we investigate the types of objects shared as well as the availability behavior of wireless users.

3.1 Analysis of objects shared by iTunes users

We analyzed the various attributes of the shared objects with an eye towards understanding whether serendipitous sharing can be fruitful.

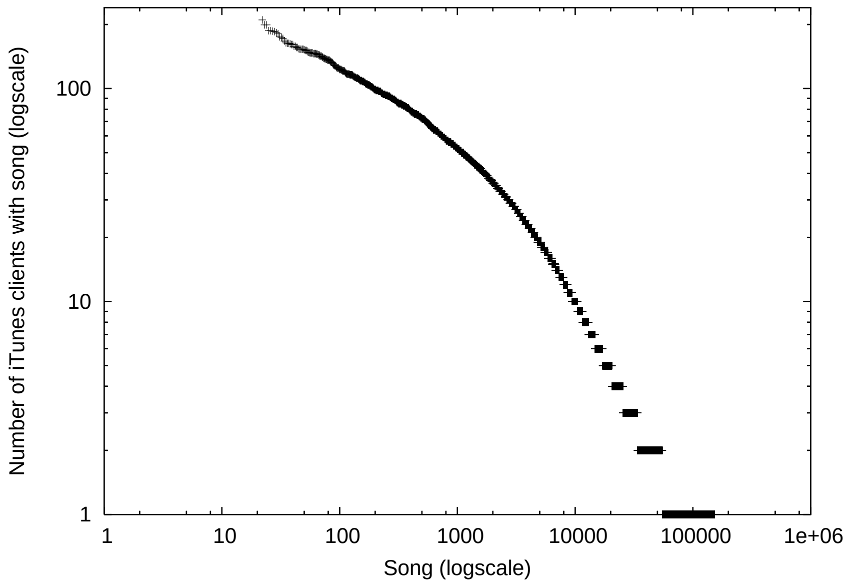
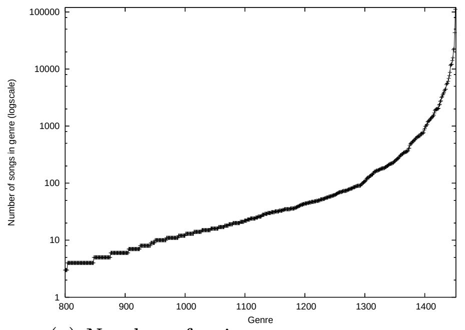


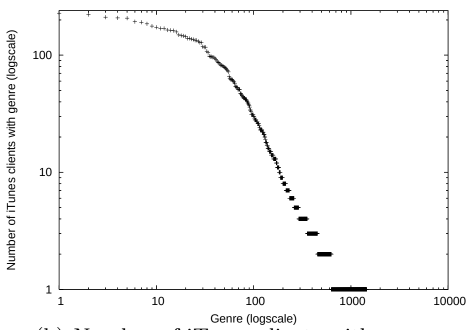
Fig. 5. Number of iTunes clients with song

3.1.1 [illegible] First we investigate the song names and the likelihood of finding a particular object among the shared songs. Unlike P2P systems such as Gnutella where users issue a query for objects of interest (which is then propagated to other nodes; matching responses are routed back to the user), iTunes users attach to a share (providing the password if the share was password protected). Once attached, the users can browse for interesting songs until either user discontinues and disconnects from sharing. Finding a song involves sequentially attaching to each available share (we explore alternative search mechanisms for serendipitous sharing in Section 3.1.4). As described in Section 2.3, we prefer complementary objects which are unavailable locally but are available in other iTunes shares. The amount of replication and the user availability (described in Section 3.2) further affect the ability to locate a particular song.

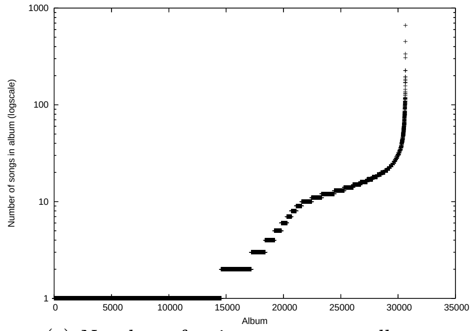
We plot the popularity distribution of the song names in all the 239 iTunes clients using a log-log scale in Figure 5. A straight line in the log-log scale denotes a pure Zipf distribution; we observed a Zipf like distribution. We also observed a similar distribution for Gnutella object annotations [Acosta and Chandra 2008a]. Since the data was collected from a single university campus, we expected more homogeneity of shared songs. 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 [illegible] appeared more than once in several clients. iTunes can potentially provide the users with a wide variety of songs (as many as 90% of the songs are unique). On the other hand, many objects were only available in a single user. Depending on the times when these users were sharing their contents (and whether these times overlap with the current user), it is highly improbable to locate objects which are available in very few nodes. The Zipfian



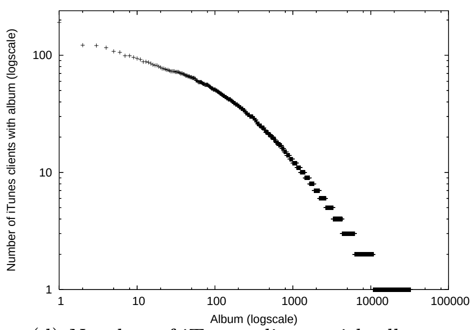
(a) Number of unique songs per genre



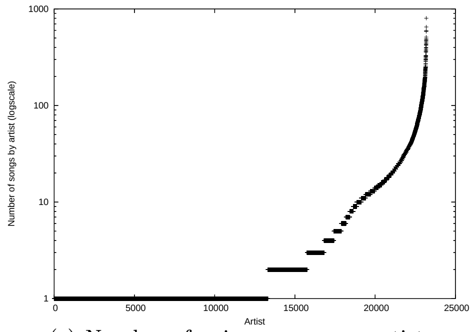
(b) Number of iTunes clients with genre



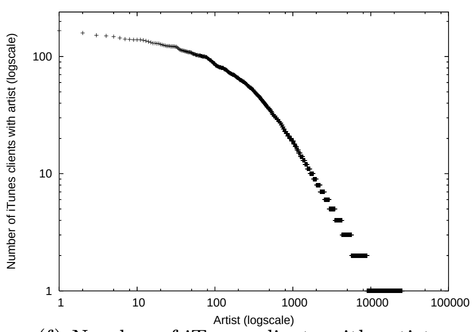
(c) Number of unique songs per album



(d) Number of iTunes clients with album



(e) Number of unique songs per artist



(f) Number of iTunes clients with artist

Fig. 6. Popularity distribution of albums, artists and genre

nature of object names meant that 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. Users might browse for songs that belong to the same genre/album/artist, while excluding particular objects that they themselves have. Next, we analyze whether such a categorization improved the availability of shared contents among iTunes users.

3.1.2 [illegible] Next, we analyze the shared objects in order to understand the efficacy of clustering them by song categories. In this model of sharing, users connect to other shares and seek all songs that belong to a certain category. We categorized songs by the same genre, album and the same artist. First, we plot the popularity distribution of the number of clients that have objects from the same category in Figures 6(b), 6(d) and 6(f), respectively. Unlike in the case of song names where we did not prefer duplicate objects, the availability of many 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 6(a), 6(c) and 6(e) respectively. Ideally, the number of songs available in categories such as albums, artists and genres should be high (to counteract for the poor availability of wireless users).

First we analyze the behavior for grouping the shared objects based on the genre. By default, iTunes is shipped with 24 genres (including [illegible]) though users are free to change the genre name. We expected that users would retain the default genres. However, from Figure 6(b), we observed that the users used 1,452 genres. Many of these genre were variations of existing genres (e.g., [illegible]) while some of them appeared to mean the same category. For example, the genres [illegible] might refer to the same category. There were 99,987 songs (18.7%) that did not have any genre. Figure 6(a) showed that many genres had tens of songs. The genres which accounted for more than 1% of the songs were: 1.01365% - [illegible], 1.02425% - [illegible], 1.22965% - [illegible], 1.28383% - [illegible], 1.30043% - [illegible], 1.40947% - [illegible], 1.58444% - [illegible] & , 1.76472% - [illegible], 2.01738% - [illegible], 2.67854% - [illegible], 2.76453% - [illegible], 2.79657% - [illegible], 3.24058% - [illegible], 3.62095% - [illegible], 5.03964% - [illegible], 5.22591% - [illegible], 10.0168% - [illegible] & [illegible] and 25.7948% - [illegible] 1219 genres (about 84%) occurred in six or less clients, potentially affecting the ability of users to successfully find them. We concluded that genre is not a good way to categorize contents because genre categorization is customized subjectively by a particular user. However the user themselves might not be aware of this personalization because from their perspective, the popularity of local songs in a certain genre creates the illusion that the particular categorization is popular in other clients as well. The user would need to search widely in order to realize that a local genre is globally unpopular.

Next we analyze the songs based on their album names. 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 6(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 name (occurring in 191 clients) was " [illegible] ", a generic album title that was created by multiple and unrelated artists. The next most popular album names were " [illegible] " by " [illegible] " and " [illegible] " by " [illegible] " which appeared in about 122 users. Typically, one expects audio albums to consist of about ten songs. Suppose a particular client has two songs from an album, we can expect to find eight more songs in other users 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). Note that currently it is not possible to know the number of songs from a particular album that are available in the other shares (without connecting to each share). Later, we propose to use bloom filters to disseminate this information (Section 3.1.4). 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.

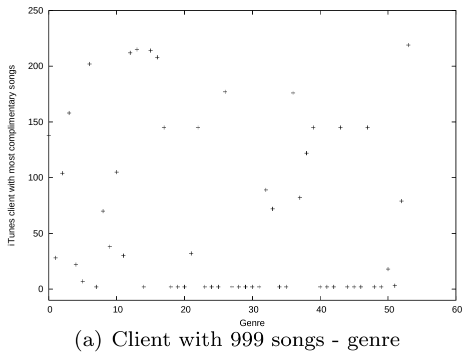
Next, we analyze the songs based on their artist names (a song can be performed by many artists). Our traces showed songs from 25,309 unique artists though 22,674 (89.5%) artists were associated with six or less clients. From Figure 6(e), we note that 13,316 (52.6%) artists had only one song in our system, clients that host these artists and searching the system to find more choices will fail because all the songs that are available among the users are already available in the local user.

To summarize, broader categorization is necessary in order to find complementary songs for serendipitous media sharing systems. Many of the categories had few objects. An object access mechanism that searches for related objects on categories that it possess should avoid searching for categories that are not popular. About 90% of these categories will not lead to related objects as the only client that hosts these objects are themselves. Searching by artist names appears promising as only 52.6% of artists had only one song in the system, the rest had many more songs.

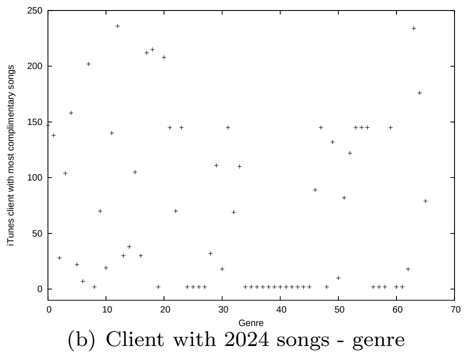
3.1.3 [illegible] Next, we investigate whether the sharing mechanism supported by current iTunes software would lead to the users finding interesting objects. Currently, the sharing is sequential; users cannot mix songs from two different clients to create a more complex playlist.

Objects of interest for a particular user depends on various factors (described in Section 2.3). First, we investigate whether connecting to a single share can provide a rich set of complementary songs based on local genre, album or artists. For these experiments, we analyzed this question for each individual client. Specifically, for each client, we identify all the genres (and albums or artists) that the user owns and for each of these genres (and albums or artists) we find the client that provides the most number of complementary songs. We define this client by counting the number of songs that the remote client has in this particular category (same genre, album, artist) that are not present in the local client. We then plot a graph between the local genres (and album or artists) and the best client that provided complementary songs for two representative clients that hosted 999 songs and 2,024 songs in Figure 7. 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 [illegible] 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 genres (and albums or artists). Hence, we believe that the iTunes sharing mechanism would require too many explicit connection switching from client to client to find many interesting objects.

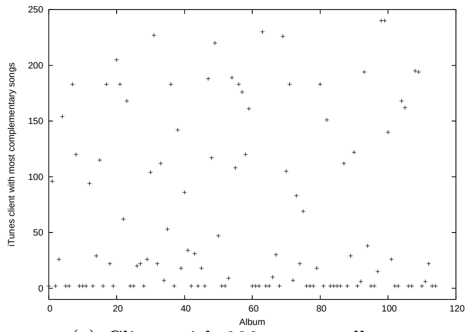
Currently, users manually switch the shared hosts with no mechanisms to give hints on choosing good peers. Without such help, the user is likely to stay with



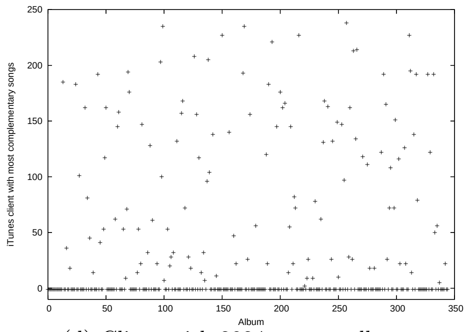
(a) Client with 999 songs - genre



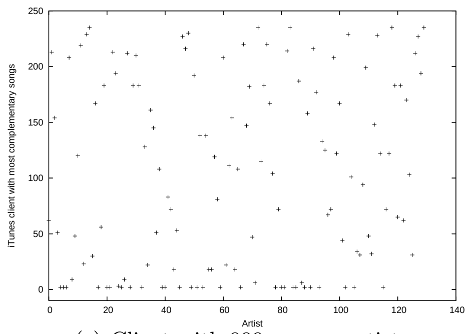
(b) Client with 2024 songs - genre



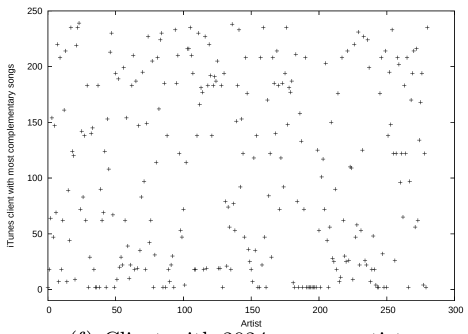
(c) Client with 999 songs - album



(d) Client with 2024 songs - album



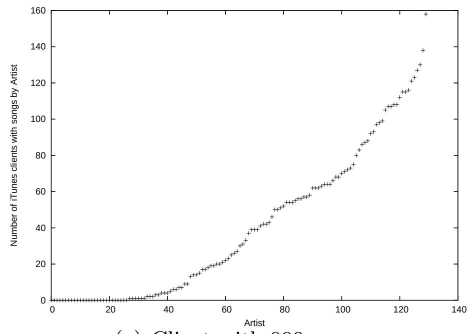
(e) Client with 999 songs - artist



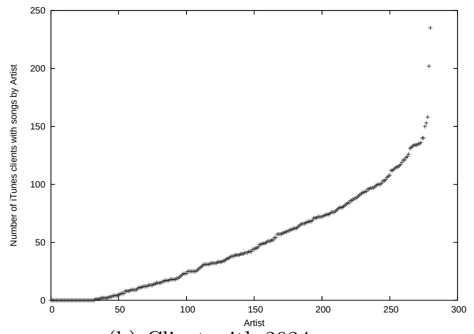
(f) Client with 2024 songs - artist

Fig. 7. Best client to host complementary object

clients which may provide a good collection for one artist (and genre or album) but not so good from another artist. We analyzed whether the system that hosts complementary songs are robust. For each artist found on the representative 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 users that host complementary songs. We note that popularity distribution depends on the particular artist; not all artists have many clients with some complementary songs.



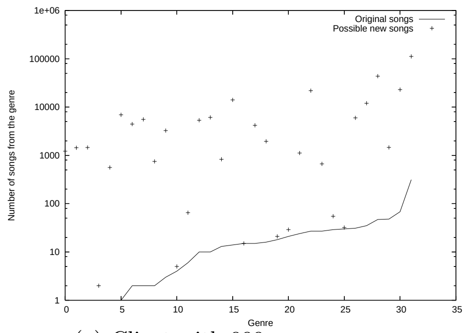
(a) Client with 999 songs



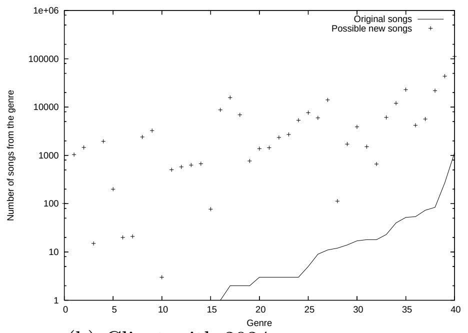
(b) Client with 2024 songs

Fig. 8. Clients that host complementary songs by artist

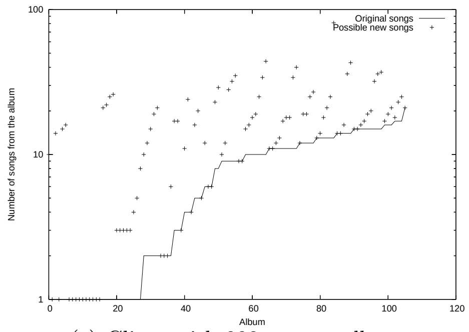
3.1.4 [illegible] So far, we have shown that iTunes users share a substantial number of songs. However, the current iTunes sharing mechanism of randomly connecting to a client and expecting to find an interesting set of related objects is not efficient. A better search mechanism for access in this environment 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 own collection. Note that the newer iTunes (version 8) supports the proprietary Genius feature that centrally aggregates the play list behavior of all iTunes users in order to generate related objects. Since many of the songs, albums and 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 (genre, album and artist) and fruitlessly search for complementary contents online. Note that about 90% of the album and artist names were associated with a single object and hence if a user already had the album and searched for new songs then they will not locate any more songs. For our analysis, we focus on the categories of songs from the same genre, album and artist. For the two typical clients investigated in the previous section, we analyze the number of songs per genre, album or artist in the particular clients collection and plot them in Figure 9. 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 a higher number of objects in other clients. 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 from the local user and the global system. For example, albums (genre 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 user to know whether a particular artist/album is worth searching for in the immediate neighborhood.



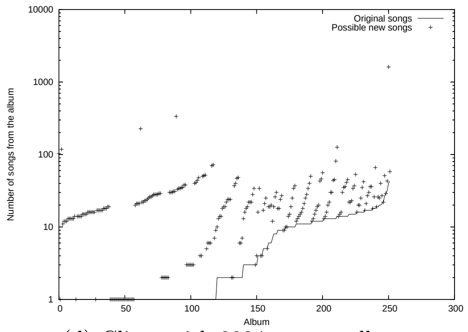
(a) Client with 999 songs - genre



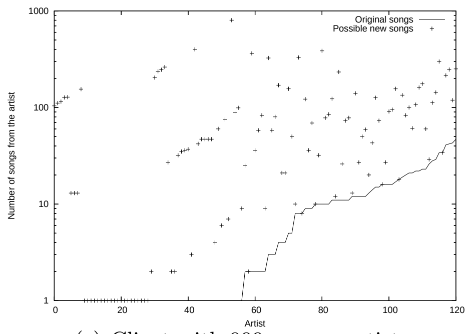
(b) Client with 2024 songs - genre



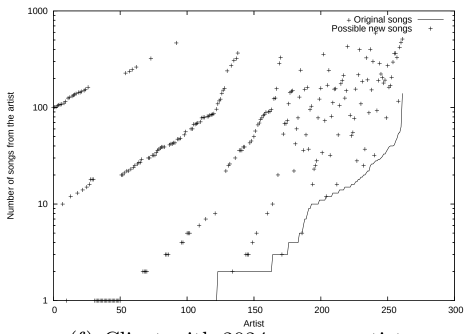
(c) Client with 999 songs - album



(d) Client with 2024 songs - album



(e) Client with 999 songs - artist



(f) Client with 2024 songs - artist

Fig. 9. Count of number of songs per category in local collection

One way to achieve this goal is to use a reactive policy that sends a bloom filter [Broder and Mitzenmacher 2004] representation of the local songs (within the category) to the remote clients and have them report back songs that were not represented in the bloom filter. 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 in the same category. Clients which host a matching category 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. However, depending on the size of the representation, bloom filters will exhibit a low false positive rate potentially missing a few complementary songs. 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 3.1.2, a significant number of songs from categories such as the 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.

A pro-active approach would push a bloom filter representing all the songs in the local client to all other users and a bloomier filter [Chazelle et al. 2004] to give the count of objects within the category to all the neighbors. The local user can then figure out how many complementary songs each client 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 user. 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 3.1.2).

For the above cases, 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 (1,500 MTU bytes). Hence, we analyzed [Manolios] 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 1,500 byte packet with 1,000 objects, the optimal number of functions was eight with a false positive rate of 0.3%. However, for 10,000 entries, the number of functions was one with a false positive rate of 56.5%. In fact, we required 15,000 bytes (over 10 packets) to reduce the false positive rate to 0.3%. In our study, users were 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 [Mitzenmacher 2001] in order to reduce the network packet sizes.

3.1.5 [illegible] In general, serendipitous sharing requires objects which are complementary to the local users collection; it also requires objects which are similar but not identical. We analyzed the songs shared by iTunes users in a campus-wide environment. We expected the user population to be more homogeneous than a community of travelers at an airport. Our analysis showed that the objects exhibit a Zipfian long tail distribution. This meant that searching for a specific object is likely to be difficult. We required mechanisms to cluster the available contents. We presented our analysis of a bloom filter based approach to cluster contents that allowed the client to access the shared contents.

3.2 Availability patterns of iTunes users

In the last section, we analyzed the attributes of objects shared by wireless users. However, serendipitous sharing also depends on the availability patterns of the various iTunes users. When few users are available, serendipitous sharing is not

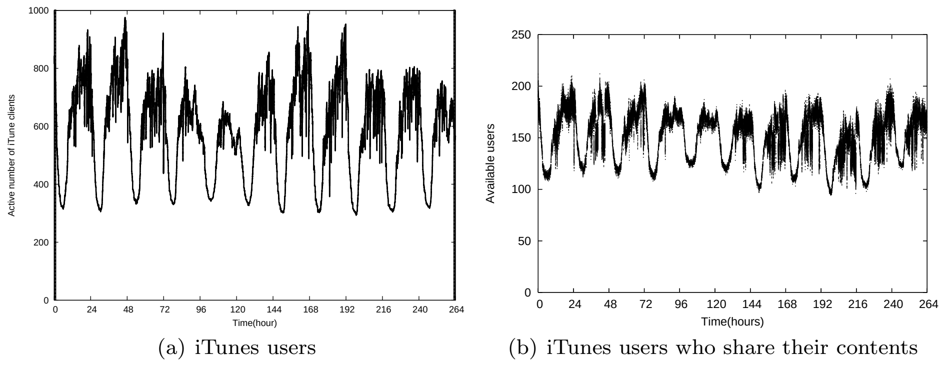


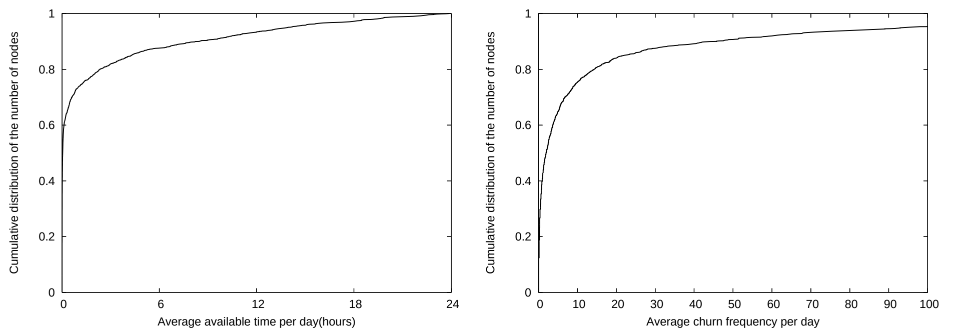
Fig. 10. Number of simultaneously available iTunes users

practical, even if the unavailable users had a vast variety of objects available for sharing. In this section, we investigate the availability patterns of iTunes users with an aim of understanding their implications for serendipitous sharing. We used the availability traces of iTunes users and durations when such users had enabled sharing from our campus (described in Section 2.1.2).

3.2.1 [illegible] We plot the number of iTunes users and the number of iTunes users that were sharing objects by the time of the day in Figure 10. The data capture began on a Tuesday. We observed that the system exhibited a diurnal behavior with less activity in the early morning hours (1 AM to 9 AM). The number of simultaneously available iTunes users varies between 350 and 1,000. Of the 1,702 unique users, between 100 and 250 (6%-15%) were simultaneously available and sharing their contents (Figure 10(b), measured using DAAP). As a medium sized university, Notre Dame's population includes 1,700 graduate students, 6,400 undergraduate students and about 1,600 faculty and administrators. During the two weeks prior to September 25, 2006, about 9,600 computers used our wireless networks. The observed number of iTunes users were a significant portion of the entire wireless user population.

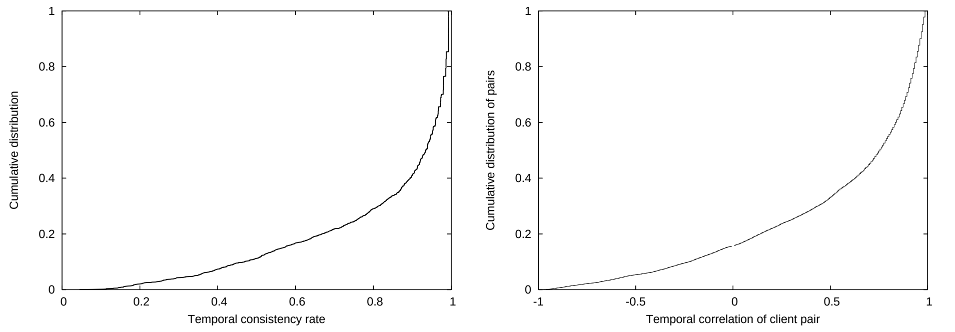
Next, we manually analyzed the traces for users that were available in the early hours (say 3:00 AM). For example, one such user was available from 1:00 AM to 9:00 AM, followed by availabilities during the day (e.g., 4:08 to 4:38 PM and 7:10-7:54 PM). Such behavior was consistent with laptop users who were mobile during the day and were sharing their contents while their laptops were left charging during the night. Prior work had also noticed similar behavior with about 30% always-on users for the most active trace from Dartmouth dataset [Henderson et al. 2004] and about 20% always-on users for the IBM [Balazinska and Castro 2003] and USC [jen Hsu and Helmy 2005] datasets.

Next, we analyzed the average user daily available duration and churn rates. The daily available durations were computed by averaging the daily available time across the entire eleven days while the daily churn frequency measured the number of available/unavailable transitions in a single day. Users may be unavailable because they went offline or because they explicitly disallowed sharing. The daily available duration gives an indication of the sharing potential and churn rates gives



(a) Daily average available time (b) Churn frequency (times user unavailable in a day)

Fig. 11. Average daily availability behavior across all users



(a) Temporal available time consistency (b) Pairwise user correlation

Fig. 12. Predictability of user availability

an indication of the user reliability. We plotted the distributions of the average daily available time and the churn frequencies in Figures 11(a) and 11(b), respectively. From Figure 11(a), we noted that the availability was poor for a large number of users. The median available duration was less than an hour. On the other hand, over 15% of the users were available for more than three hours in a single day and 13% of users were available for more than six hours on average in a single day. Figure 11(b) showed that the median churn rate was about one. About 20% of the users exhibited large churn rates, in excess of ten times in a single day.

3.2.2 [illegible] So far, we noted that the availability of wireless users was poor. Even if the users were not available all the time, serendipitous sharing systems can perform well when sharing users were simultaneously online. We analyzed such behavior along two axes: whether a given user became temporally available and unavailable in a predictable fashion and whether a community of users always occurred as a correlated group.

3.2.2.1 [illegible] We investigated whether users were temporally consistent. Suppose a significant percentage of the users were available at (say) 10 AM. One can then build a sharing system that will give good consistency during this time on every day (when other users are also simultaneously online). On the other hand, if we noticed that a significant number of users were not consistently available at (say) 10 AM, then one can safely ignore this time period. Even though one cannot build a sharing system at this particular time, it may not matter because there were no other users who would care about the non-availability of storage at this time.

Next, we computed the rate of available time consistency as follows: we defined consistency at a specific hour by the metric that the user will either be consistently available or unavailable on all the eleven days. For example, if either the user was unavailable on all eleven days or the user was unavailable on all eleven days at a specific time, we computed the consistency at that time as one. If the user was available for half the time and unavailable for the other half, then the consistency was zero. We computed the consistency values for the user in steps of one hour for the entire 24 hour day and normalized it by dividing by 24. We plotted a cumulative distribution of the rate of available time consistency in Figure 12(a). A value of one indicated that all users were consistent (always available at all times or unavailable at all times), whereas a value of zero implied that users were equally likely to be either available or unavailable with no consistent way to predict their behavior. From Figure 12(a), we noted that 90% of the users had consistency values of over 0.7. However, only 5% of the users had values of 0.4 or lower. These values suggest that, even though there may not be many users who are available at all times, a large number of users were predictable in terms of times that they were available (or unavailable) and a small number of users consistently form a recurring group. This behavior has significant implications for developing serendipitous sharing.

3.2.2.2 [illegible] Next, we plotted the cumulative distribution of the pairwise user correlation value for all the user pairs in Figure 12(b). This function was previously described by [Bolosky et al. 2000]. We computed the pairwise user correlation as follows: add one when two users were either simultaneously available or unavailable and subtract one when only one of the two users were simultaneously available. We sampled the system every hour. The results were normalized by dividing by the total sample count. We preferred values of one as it suggested that the pair of users were either both available or unavailable. Note that a value of zero suggests that the behavior of the pair of users was unpredictable. From Figure 12(b), we noted that over 50% of the user pairs had a correlation value of 0.7. Our users showed higher correlation than corporate desktops [Bolosky et al. 2000], which observed values of 0.5 for 50% of the user pairs.

3.2.3 [illegible] In the previous sections, we analyzed various aspects of wireless user availability. Next, we investigate the implications of user availability for serendipitous media sharing. For our analysis, we randomly choose two, ten as well as 25 users from our availability trace. We assume that each of these users had a copy of the same song (judged by the name) and were sharing this object every time that they were online. We repeated this experiment by using 1,000 different user groups and plot the average values of temporal object availability for serendipitous media sharing in Figure 13. Note

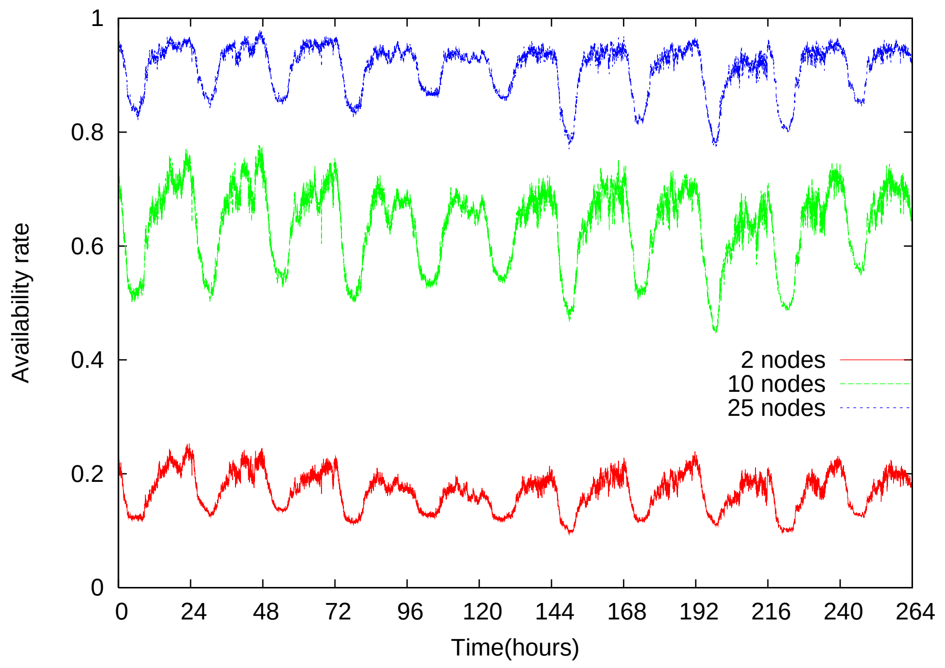


Fig. 13. Availability rates for everyone

that the object is unavailable when none of the replicas are available, while the object is considered to be available if at least one of the users was available.

From Figure 13, we note that the object availability suffered during the durations when few users were available (e.g., early morning). Since few users were available during these hours, the effective availability could be high. The object availability also changed with the replication rates. The objects were available for about 20% of the time for an object that was replicated twice with an availability rate of 95% for using 25 replicas. On the other hand, iTunes song names were observed to follow a Zipfian distribution (Section 3.1.1). For serendipitous media sharing to be fruitful, the number of copies of the object should increase. Currently, iTunes does not retain songs that are played from other shares. One way to improve the replication factor is to allow the users to retain copies of objects that they listen to. This mechanism is especially attractive when the song DRM (e.g., podcasts) might not restrict such profuse propagation. Another approach is to use an infrastructure based storage for limited durations. Next, we investigate such a mechanism.

3.2.4 [illegible] One technique to improve object availability is to replicate the object and store them in a highly available server. However, the DRM requirements might not allow replication for unlimited durations. However, limiting the duration of storage might allow the content providers to permit such replication. For example, Microsoft Zune allows the players to directly [illegible] their songs to other users who are within wireless range. Microsoft limited the usability of these objects to be playable for

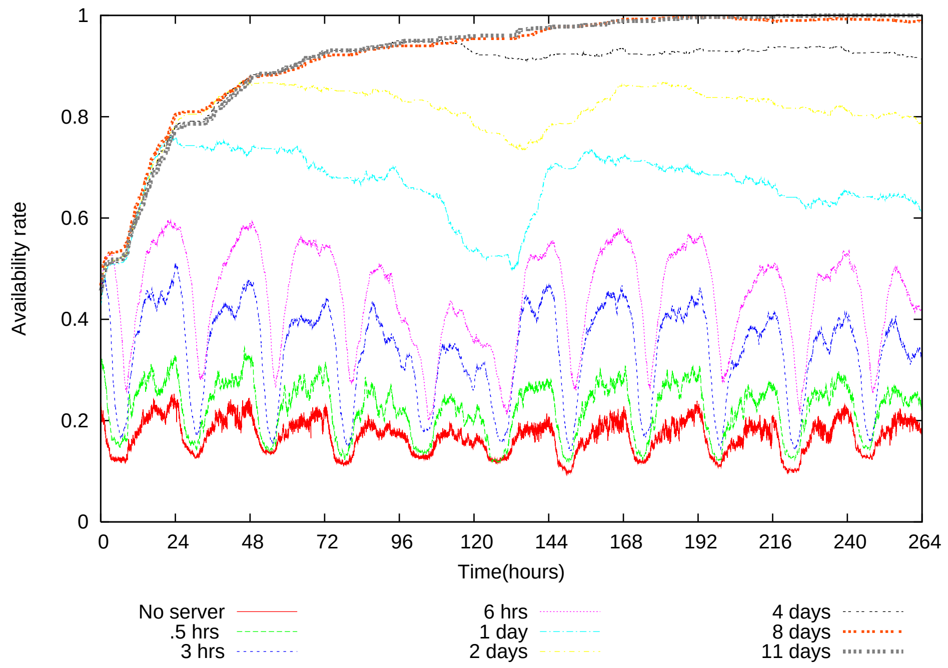


Fig. 14. Improving object availability using limited durations of infrastructure storage

up to three times in three days. Also, these objects cannot be propagated further to other users. Microsoft has since relaxed on this three day restriction. We replicated this behavior by storing the object in a server for a limited duration (while forbidding the wireless users themselves from retaining the songs). While online, users uploaded their contents to the server. The median share size in our traces was 10 GB (Section 2.1.1; using an 802.11n wireless network (typical throughput of 100 Mbps), it would take about 15 minutes to upload the entire collection. Depending on the wireless network capacity, the client might not be able to replicate all its contents before going offline. In that case, only objects which were successfully uploaded will enjoy the benefits of the server storage. Our goal was to understand the least amount of storage duration required to improve object availability.

We varied the storage durations, repeated the experiments for a thousand different users and reported the average behavior in Figure 14. From Figure 14, we note that a single copy and server storage for three hours can improve the availability from 10% to 25%. We achieve even higher availability using longer durations.

4. RELATED WORK

Various P2P systems had been developed to address challenges in large scale data distribution and management. Unstructured P2P systems such as Gnutella [gnutella-v04] organized topologies using local information. Structured overlays such as Chord [Stoica et al. 2001] built a distributed hash table to store and locate contents. Saroiu et al. [Saroiu et al. 2002; Gummadi et al. 2003] analyzed the behavior of Gnutella overlays. Zhao et al. [Zhao et al. 2006] and Acosta et al. [Acosta and Chandra 2008a] analyzed the objects stored in Gnutella networks. Our work focuses on local sharing that allows for system choices that are not always possible in large scale systems.

In our earlier work, we [Chandra and Yu 2007] analyzed the serendipitous sharing mechanism. We [Yu and Chandra 2008] also investigated the feasibility of a targeted mechanism to distribute lecture videos to students registered in a particular course.

Farsite [Adya et al. 2002] is the closest project to this research in terms of operating 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 [Kubiatowicz et al. 2000] were designed for. Based on the observed availability, Farsite developed distributed file systems [Adya et al. 2002] accessible to corporate users. Our observed availability cannot sustain a file system like sharing semantics.

There is a danger that P2P systems, including the system described in this paper will lead to illegal object sharing. RIAA [Graham 2006] 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. Kirovski et al. [Kirovski and Jain 2006] are addressing the economics of allowing users to distribute (and sell) contents to immediate neighbors.

A number of prior efforts analyzed the behavior of users under a variety of application and networking scenarios. Farsite [Bolosky et al. 2000] analyzed the behavior of wired corporate desktops using network ping messages. Balazinska et al. [Balazinska and Castro 2003] analyzed the behavior of corporate wireless users using access point (AP) SNMP probes. A number of prior efforts [Tang and Baker 2000; Kotz and Essien 2002; Henderson et al. 2004] monitored the WLAN network in a university setting using packet traces and AP SNMP probes. Hsu et al. [jen Hsu and Helmy 2005] presented a comprehensive analysis of the user mobility behavior across four different university campuses using AP logs. We analyzed higher level availability of university users. Data link layer mobility was not captured; any user who migrated across APs were considered to be continuously online even though they associated and disassociated with multiple access points.

Recent work had focused on scenarios which lacked the storage and network infrastructure, necessitating an understanding of the spatial behavior of users. Hui et al. [Hui et al. 2005] described the notion of pocket switched networks (PSN) wherein nodes utilized human mobility as well as infrastructure to propagate data between mobile users. They captured user contact patterns by distributing their hardware capture device to about fifty-four participants at the INFOCOM 2005 conference. Chaintreau et al. [Chaintreau et al. 2006] further investigated opportunistic forwarding algorithms using the captured contact information of volunteer conference attendees. Hui et al. [Hui and Crowcroft 2007] analyzed the routing behavior of these contact based PSNs that also utilized an infrastructure to forward the queries. Similarly, Mickens et al. [Mickens and Noble 2005] introduced a probabilistic queuing framework to spread computer viruses that launched a proximity attack between mobile users. More recently, Yan et al. [Yan et al. 2007] described the node spatial and temporal distribution parameters required for propagating worms using blue tooth networks. Song et al. [Song and Kotz 2007] used the AP records to collect contact patterns among wireless users. They observed that the epidemic propagation can be unacceptably long, especially among casual users (some users might never meet each other in the future). For our system, we assumed the availability of a wireless infrastructure; either via APs or by using ad hoc/mesh networks. Given the prevalence of wireless APs and mesh networks, such an assumption is not unrealistic. This assumption allowed us to ignore the spatial mobility patterns of the users; we considered any two nodes that were online anywhere on campus to be available. We expected our node availability to be far better than that was observed using user vicinity contact measurements. However, we observed limited system sharing performance. This places serious doubts on the viability of a system that provided its own wireless and storage infrastructure.

Zhuang et al. [Zhuang et al. 2003] allowed a mobile user to hand off packets to other mobile users through an Internet indirection infrastructure [Stoica et al. 2004]. They assumed the presence of an overlay infrastructure which was able to store these messages till they were explicitly deleted. In Section 3.2.4, we improved our sharing system performance by storing contents in a storage infrastructure. Our work seeks to limit the duration of reliance on such infrastructure.

5. DISCUSSION AND FUTURE WORK

Peer-to-peer systems have emerged as a popular mechanism to share contents among a large number of users. In this paper, we investigated the behavior of sharing systems built among a small community of users. Popular systems such as Apple iTunes and Microsoft Zune use variations of this basic mechanism. We investigated the kinds of objects stored among a local community of university wireless users. We showed that the contents should be complementary for fruitful sharing; the contents should be related to local content and yet not be identical. However, we observed that the objects exhibited Zipf like long tail distribution regardless of analysis along annotation axes such as genre, album and artist. This meant that serendipitous sharing mechanisms that expect to find interesting objects in their community need to broaden the scope of objects into larger classes of objects. Users are unlikely to find specific objects that they already do not possess. On the other hand, the poor user availability forces these systems to highly replicate the objects. The performance can also be improved through modest reliance on storage infrastructure. Our work makes important contributions for community based sharing systems while also offering a cautious note regarding specific sharing mechanisms.

ACKNOWLEDGMENTS

Supported in part by the U.S. National Science Foundation (CNS-0447671).

REFERENCES

Acosta, W. and Chandra, S. 2007. Trace driven analysis of the long term evolution of gnutella peer-to-peer traffic. In *the eighth Passive and Active Measurement conference (PAM 2007)*. Louvain-la-neuve, Belgium.

Acosta, W. and Chandra, S. 2008a. On the need for query-centric unstructured peer-to-peer overlays. In *IEEE Hot Topics in Peer-to-Peer Systems (HotP2P '08)*. Miami, FL.

Acosta, W. and Chandra, S. 2008b. Understanding the practical limits of the gnutella p2p system: An analysis of query terms and object name distributions. In *ACM/SPIE: Multimedia Computing and Networking (MMCN'08)*. Vol. 6818. San Jose, CA, 681806–1 – 681806–12.

Adya, A., Bolosky, W. J., Castro, M., Cermak, G., Chaiken, R., Douceur, J. R., Howell, J., Lorch, J. R., Theimer, M., and Wattenhofer, R. P. 2002. Farsite: federated, available, and reliable storage for an incompletely trusted environment. *SIGOPS Oper. Syst. Rev. 36,* SI, 1–14.

Aubrey, S. 2003. Adventures in higher education: ipod envy. The Wesleyan Argus. `http://www.wesleyan.edu/argus/archives/nov042003/dateyear/w1.html`.

Balazinska, M. and Castro, P. 2003. Characterizing mobility and network usage in a corporate wireless local-area network. In *MobiSys '03*. 303–316.

Bolosky, W. J., Douceur, J. R., Ely, D., and Theimer, M. 2000. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *ACM SIGMETRICS*. 34–43.

Broder, A. and Mitzenmacher, M. 2004. Network Applications of Bloom Filters: A Survey. *Internet Mathematics 1,* 4, 485–509.

Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. 2006. Impact of human mobility on the design of opportunistic forwarding algorithms. In *INFOCOM 2006*. Barcelona, Spain, 1–13.

Chandra, S. and Yu, X. 2007. Share with thy neighbors. In *ACM/SPIE Multimedia Computing and Networking (MMCN 2007)*. San Jose, CA.

Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. 2003. Making gnutella-like p2p systems scalable. In *Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*. 407–418.

Chazelle, B., Kilian, J., Rubinfeld, R., and Tal, A. 2004. 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*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 30–39.

Davies, C. Applerecords. `www.cdavies.org/applerecords.html`.

dnssd. Dns service discovery. `http://www.dns-sd.org/`.

gnutella-v04. The gnutella protocol specification v0.4. `http://dss.clip2.com/GnutellaProtocol04.pdf`.

Graham, J. 2006. Riaa chief says illegal song-sharing 'contained'. USA TODAY.

Gummadi, K. P., Dunn, R. J., Saroiu, S., Gribble, S. D., Levy, H. M., and Zahorjan, J. 2003. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 314–329.

Guttman, E. 2001. Autoconfiguration for ip networking: Enabling local communication. *IEEE Internet Computing 5,* 3, 81–86.

Henderson, T., Kotz, D., and Abyzov, I. 2004. The changing usage of a mature campus-wide wireless network. In *MobiCom '04*. 187–201.

Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., and Diot, C. 2005. Pocket switched networks and human mobility in conference environments. In *2005 ACM SIGCOMM workshop on Delay-tolerant networking*. 244–251.

Hui, P. and Crowcroft, J. 2007. How small labels create big improvements. In *IEEE International Conference on Pervasive Computing and Communications Workshops*. 65–70.

jen Hsu, W. and Helmy, A. 2005. Impact: Investigation of mobile-user patterns across university campuses using wlan trace analysis. Tech. rep., USC. July.

Kirovski, D. and Jain, K. 2006. Off-line economies for digital media. In *Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '06)*. New port, RI, 118–123.

Kotz, D. and Essien, K. 2002. Analysis of a campus-wide wireless network. In *ACM MobiCom '02*. 107–118.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., and Zhao, B. 2000. Oceanstore: an architecture for global-scale persistent storage. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*. 190–201.

Lovitt, R. 2008. In-flight internet access takes off. `http://msnbc.msn.com/id/25463095/`.

Manolios, P. Bloom filter calculator. `http://www-static.cc.gatech.edu/~manolios/bloom-filters/calculator.html`.

Massoulie, L., Merrer, E. L., Kermarrec, A.-M., and Ganesh, A. 2006. Peer counting and sampling in overlay networks: random walk methods. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. 123–132.

Mickens, J. W. and Noble, B. D. 2005. Modeling epidemic spreading in mobile environments. In *4th ACM workshop on Wireless security (WiSe '05)*. 77–86.

Missouri 2009. Audio-video player with web browser requirement undergraduate program. `http://journalism.missouri.edu/undergraduate/web-media-player.html`.

Mitzenmacher, M. 2001. Compressed bloom filters. In *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. 144–150.

Rowstron, A. and Drushel, P. 2001. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM Conference on Distributed Systems Platforms (Middleware 2001)*. Heidelberg, Germany, 329–350.

Saroiu, S., Gummadi, P. K., and Gribble, S. D. 2002. A measurement study of peer-to-peer file sharing systems. In *MMCN 2002*.

Schiller, P. 2009. Macworld expo 2009 keynote.

Song, L. and Kotz, D. 2007. Evaluating opportunistic routing protocols with large realistic contact traces. In *ACM MobiCom workshop on Challenged Networks (CHANTS 2007)*.

Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. 2004. Internet indirection infrastructure. *IEEE/ACM Trans. Netw. 12,* 2, 205–218.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. 149–160.

Stutzbach, D., Rejaie, R., and Sen, S. 2007. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *IEEE/ACM Transactions on Networking*.

Tang, D. and Baker, M. 2000. Analysis of a local-area wireless network. In *ACM Mobicom '00*. 1–10.

Timiraos, N. 2006. Free, legal and ignored. Wall Street Journal, July 6, 2006 - Page B1.

Voida, A., Grinter, R. E., Ducheneaut, N., Edwards, W. K., and Newman, M. W. 2005. Listening in: practices surrounding itunes music sharing. In *ACM CHI '05*. 191–200.

Yan, G., Flores, H. D., Cuellar, L., Hengartner, N., Eidenbenz, S., and Vu, V. 2007. Bluetooth worm propagation: mobility pattern matters! In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*. 32–44.

Yu, X. and Chandra, S. 2008. Campus-wide asynchronous lecture distribution using wireless laptops (short paper). In *ACM/SPIE: Multimedia Computing and Networking (MMCN'08)*. Vol. 6818. San Jose, CA, 68180M–1 – 68180M–8.

Zaharia, M. A., Chandel, A., Saroiu, S., and Keshav, S. 2007. Finding content in file-sharing networks when you can't even spell. In *6th Workshop on Peer-to-Peer Systems (IPTPS'07)*. Bellevue, WA.

Zhao, S., Stutzbach, D., and Rejaie, R. 2006. Characterizing files in the modern gnutella network: A measurement study. In *Proceedings of SPIE/ACM Multimedia Computing and Networking*. Vol. 6071. San Jose, CA.

Zhuang, S., Lai, K., Stoica, I., Katz, R., and Shenker, S. 2003. Host mobility using an internet indirection infrastructure. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, New York, NY, USA, 129–144.

Received October 2008; revised May 2009; accepted October 2009