Pollution in P2P File Sharing Systems

Jian Liang

Department of Computer and
Information Science,
Polytechnic University,
Brooklyn, NY
Email: jliang@cis.poly.edu

Rakesh Kumar

Department of Electrical and
Computer Engineering,
Polytechnic University,
Brooklyn, NY
Email: rkumar04@utopia.poly.edu

Yongjian Xi

Department of Computer and
Information Science,
Polytechnic University,
Brooklyn, NY
Email: yxi@cis.poly.edu

Keith W. Ross

Department of Computer and
Information Science,
Polytechnic University,
Brooklyn, NY
Email: ross@poly.edu
Phone : 1-718-260-3859

Abstract— One way to combat P2P file sharing of copyrighted content is to deposit into the file sharing systems large volumes of polluted files. Without taking sides in the file sharing debate, in this paper we undertake a measurement study of the nature and magnitude of pollution in KaZaA, currently the most popular P2P file sharing system. We develop a crawling platform which crawls the majority of the KaZaA 20,000+ supernodes in less than 60 minutes. From the raw data gathered by the crawler for popular audio content, we obtain statistics on the number of unique versions and copies available in a 24-hour period. We develop an automated procedure to detect whether a given version is polluted or not, and we show that the probabilities of false positives and negatives of the detection procedure are very small. We use the data from the crawler and our pollution detection algorithm to determine the fraction of versions and fraction of copies that are polluted for several recent and old songs. We observe that pollution is pervasive for recent popular songs. We also identify and describe a number of anti-pollution mechanisms.

Keywords— Network Measurements.

I. INTRODUCTION

By many measures, file sharing is the most important application in the Internet today. For example, on a typical day, KaZaA - currently the most popular file-sharing application - has more than 3 million active users sharing over 5,000 terabytes of content. On the University of Washington campus network in June 2002, KaZaA consumed approximately 37% of all TCP traffic, which was more than twice the Web traffic on the same campus at the same time [1].

But file sharing is not only having an important impact on Internet usage and traffic; it is also profoundly impacting sales in the music and video recording industries. For example, in a recent study, Forrester estimates that the music industry lost over $700 million in CD sales in 2003 due to illicit sharing of copyrighted songs in P2P file sharing systems [2]. Each week there are more than one billion downloads of music files, and over 60 million Americans have downloaded music [3] [4] .

Because of the potential of huge financial losses, the music industry has attempted to throttle P2P file sharing activity on three distinct fronts. First, it has taken many of the file-sharing companies to court for copyright infringement. This approach was successful in 2001, when the US courts effectively shut down the leading file sharing application, Napster, a US-based company with a centralized architecture for file location [5]. However, this approach has had little success at curtailing KaZaA, for which it is more difficult to simply "pull the plug" due to its highly decentralized architecture and to its elusive international corporate structure. The second front has been to prosecute the individual users for copyright infringement, which by some estimates has decreased illicit file sharing by 20%. However, file sharing remains rampant in the Internet, as it is difficult to prosecute millions of "small" users, particularly when they are scattered across the globe. The music industry's third front for throttling file sharing is to actually sabotage the P2P file sharing systems. This approach has received relatively little press to date but - as we shall demonstrate in this paper - is currently being deployed on a grand scale.

One sabotage technique that is particularly prevalent today is that of **pollution**. Here, a "pollution company" first tampers with copyrighted content with the intention of rendering the content unusable. It then deposits the tampered content in large volumes in the P2P network. Unable to distinguish polluted files from unpolluted files, unsuspecting users download the files into their own file-sharing folders, from which other users download the polluted files. In this manner, the polluted copies of a given song spread through the file-sharing system, and the number of polluted copies can eventually exceed the number of clean copies of a given song. The goal of the pollution company is to trick users into frequently downloading polluted copies; users may then become frustrated and abandon P2P file sharing.

One such pollution company is Overpeer [6] [7]. Overpeer works with major record labels, film studios, television networks, and game publishers to pollute P2P networks. For example, when a recording company is on the verge of releasing a song that will likely be popular, the record company might pay a pollution company to spread bogus copies of the song through one or more P2P networks. The approach is described in Overpeer's US patent applications [7][8]. A similar approach is described in a recent patent application from the University of Tulsa [9]. The patent

describes cooperative scanning, manufacturing, sharing and supervisory control software to share decoy (that is, polluted) media at a volume that renders media search ineffectual [9]. Retspan is yet another example of company in the business of spreading polluted content in P2P systems [10].

In this paper we undertake a detailed measurement study of the nature and magnitude of pollution in KaZaA, currently the most popular P2P file sharing system. We emphasize that the purpose of this paper is not to take sides on the P2P file-sharing debate nor to condone nor to condemn pollution. The goal instead is to understand P2P pollution, how pervasive it is currently in P2P networks, how quickly it spreads, and to identify measures for countering P2P pollution attacks. We will see that pollution is indeed pervasive, with more than 50% of the copies of many popular recent songs being polluted in KaZaA today. Because P2P file sharing is having a major impact on Internet traffic and usage, it is important to gain deep insights into P2P pollution, which is now a central part of the P2P landscape.

The contributions of this paper include:

- We developed a powerful crawling system which crawls the majority of the KaZaA 20,000+ supernodes in less than 60 minutes. Developing such a crawler is highly challenging since KaZaA uses a proprietary protocol with most of its signaling messages being encrypted.
- From the raw data collected by the crawler, we obtained statistics for popular audio content on the number of unique versions and copies available in a 24-hour period. For a given song, we find that the number the copies versus the version number typically follows a Zipf distribution.
- In order to estimate pollution levels, we developed an automated processing procedure to detect whether a given version is polluted or not (which does not involve listening to the song). For the current pollution attacks, we show that the probabilities of false positives and false negatives of the detection procedure are very small.
- We used the data from the crawler and our pollution detection algorithm to determine the fraction of versions and fraction of copies that are polluted for several popular songs. We observe that pollution is pervasive for recent popular songs and hardly significant for older songs.
- We used our crawler to study the evolution of versions, copies, and pollution in KaZaA for a period of 19 days.
- We use or measurement data to show that the KaZaA rating system is ineffective for identifying polluted files.
- Finally, we identify and describe a number of anti-pollution mechanisms. These mechanisms fall into two categories: schemes that require the user to download a portion of the file before declaring the file polluted; and those which do not require any user downloading.

II. CLASSIFICATION OF P2P POLLUTION

Depending on the strategies taken to pollute content, pollution in file-sharing systems can be classified into two major categories.

- **Content Pollution:** This is currently the more common form of pollution. The polluting party targets a particular digital recording (e.g., song or video). It then manufactures decoys for the recording by modifying it in one or more ways, including replacing all or part of the content with white noise, cutting the duration, shuffling blocks of bytes within the digital recording, inserting warnings of the illegality of file sharing in the recording, and inserting advertisements. We observed that today a popular pollution technique is to insert tens of seconds of undecodable white noise into the middle of the song.
- **Metadata Pollution:** The other strategy is to not tamper with the digital recordings themselves but instead tamper with metadata. This often involves taking an older recording, whose copyright has expired, and changing its song title, album title, and artist name to that of the targeted recently-released recording. Thus, when a user requests the target recording, the user will mistakenly obtain a different recording.

We emphasize that these pollution schemes currently work well because there is a lack of good media matching systems in P2P file sharing. We discuss more about strategies for countering pollution in Section VI.

We can also classify pollution as **intentional** and **unintentional**. A pollution company intentionally creates polluted versions of files, using the content and metadata pollution techniques described above. But users often accidentally create damaged files and inject them into P2P file sharing systems. For example, a user may "rip" a song from a CD, inadvertently truncate the song, and then make available the truncated song in the P2P file-sharing system. Or a user may record the song from the radio and accidentally pick up the disk-jockey's voice at the beginning or end of the song. We refer to files which have been inadvertently corrupted by user error as unintentional pollution. Finally, we remark that certain parties sometimes make minor modifications in recordings which are hardly noticeable. For example, we have observed that to reduce a song's air time, a radio station may eliminate a long, repetitive tail of the song or even slightly accelerate its playback. A user can then record and distribute the slightly-tampered song in a P2P file sharing system. The songs investigated in this study are listed in Table 1. (In Section IV we explain why we chose these particular songs.)

SONG TITLE	ARTIST	LABEL	RELEASE DATE	CD DURATION (secs)
Naughty Girl	Beyonce	Columbia/Sony music	Jun 24, 2003	208
Ocean Avenue	Yellowcard	Capitol/EMI	July 22, 2003	198
Where is the love?	Black Eyed Peas	Interscope/ Universal Music	Aug 12, 2003	274
Hey Ya	OutKast	Arista/BMG	Sep 15, 2003	235
Toxic	Britney Spears	Jive/Zomba	Nov 18, 2003	198
Tipsy	J-Kwon	So So Def Records/Sony Music	Jan 26, 2004	243
My Band	D12	Interscope/ Universal Music	Mar 11, 2004	299

TABLE I

SONGS INVESTIGATED FOR POLLUTION

III. THE KaZaA CRAWLING SYSTEM

To gather raw data about versions, copies, and pollution levels in P2P systems, we developed and deployed a farm of multi-threaded crawling nodes, which we call the **KaZaA Crawling Platform**. This system crawls through virtually all of the 30,000+ KaZaA supernodes in 15-60 minutes. Furthermore, it is scalable in that the crawling time is inversely proportional to the number of Linux boxes in the platform.

A crawling system was previously developed for the Gnutella P2P network [11]. Developing a crawling system for KaZaA is significantly more challenging for two reasons. First, KaZaA is 10-100 times larger than Gnutella, both in terms of the number of peers and traffic. Second, and more importantly, the Gnutella protocol is in the public domain, whereas the KaZaA protocol is proprietary with little information available to the research community about how it operates. Thus, to develop the KaZaA Crawling Platform, we first had to undertake a measurement and reverse engineering project to understand how the KaZaA system operates [12].

A. Overview of KaZaA Design

To present the KaZaA Crawling System and our experimental methodology, we first need to summarize how KaZaA works. Our focus here is on the aspects of KaZaA that are relevant to the KaZaA Crawling System. A more complete description is available in [12].

The KaZaA system contains files available for file sharing. These files include audio mpegs, videos, and executables including games. Each file in the system includes **metadata**. A file's metadata includes the **file name**, the **file size** and **file descriptors**. For music, a file's file descriptors typically include song title, artist, album, and user-supplied keywords. The file descriptors are used for keyword matches during querying.

The KaZaA software installed and executed on the peers is called the **KaZaA Media Desktop (KMD)**. The KMD enables a peer to download files directly from other peers, upload files directly to other peers, and query for content stored in the other peers. For each file in the KaZaA shared folder, the KMD determines the file's **ContentHash**, which is a proprietary signature taken over the entire file, including the file's metadata. The ContentHash plays a central role in the KaZaA design. In the most recent version of KaZaA, the ContentHash is the only identifier used to identify a file when requesting a download. If a download from a specific peer fails, the ContentHash enables the KaZaA client to search for the specific file automatically, without issuing a new keyword query.

In addition to KaZaA, Grokster and iMesh are two other clients that currently participate in the FastTrack overlay network. All three clients are licensed by Sharman Networks, Inc and use the same protocol as KaZaA. Many users today also use KaZaA-Lite [13], an unofficial copy of the KMD, rather than the KaZaA client (KMD) distributed by Sharman. Each KaZaA-Lite client emulates Sharman's KMD and participates in the same KaZaA network. When we say KaZaA, we are actually referring to the FastTrack network and all of its clients.

Unlike Napster, KaZaA is decentralized and does not maintain an always-on, centralized index for tracking the location of files. As shown in Figure 1, KaZaA has two classes of peers, Ordinary Nodes (ONs) and Super Nodes (SNs). SNs have greater responsibilities and are typically more powerful than the ONs with respect to availability, Internet connection bandwidth and processing power. When an ON launches the KaZaA application, the ON establishes a TCP connection with a SN, thereby becoming a "child" of that SN. The ON then uploads to the SN the metadata and ContentHashes for the files it is sharing. This allows the SN to maintain a **local index** which includes ContentHashes and file descriptors for all the files its children are sharing along with the corresponding IP addresses of the ONs holding the particular files. In this way, each SN becomes a mini Napster-like hub. But in contrast with Napster, a SN is not a dedicated server (or server farm); instead, it is a peer belonging to an individual user. As shown in Figure 1, each SN also maintains long-lived TCP connections with other SNs, creating an overlay network among the SNs.

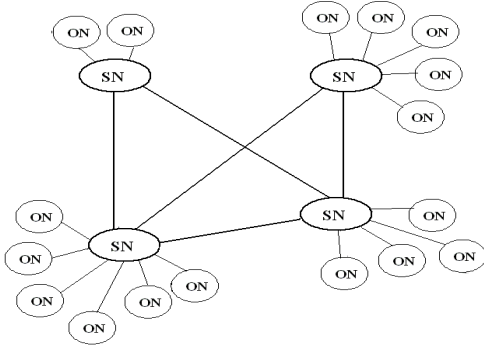When a user wants to find files, the user's ON sends a query with keywords over the TCP connection to its

Fig. 1. *Supernode and Ordinary nodes in KaZaA network*

SN. For each match in its local index, the SN returns the metadata and IP addresses corresponding to the match. When a SN receives a query, it may flood the query over the overlay network to one or more of the SNs to which it is connected. A given query will in general visit a small subset of the SNs, and hence will obtain the metadata information of a small subset of all the ONs.

As part of the signalling traffic, KaZaA nodes frequently exchange with each other lists of supernodes. For example, when an ON connects with a SN, the SN immediately pushes to the ON a **supernode refresh list**, which consists of the IP addresses of up to 200 SNs. Each ON maintains a cache of up 200 SNs whereas SNs appear to maintain a cache of thousands of SNs. When a peer A (ON or SN) receives a supernode refresh list from another peer B, peer A will typically purge some of the entries from its cache and add entries sent by peer B. By frequently exchanging supernode refresh lists, nodes maintain up-to-date lists of active SNs.

B. The KaZaA Crawling Platform Architecture

The KaZaA Crawling Platform is shown in Figure 2. It consists of a **process manager**, a **measurement database**, and n **crawling nodes**. At the core of the system are the n crawling nodes, each implemented in its own Linux box. In our current deployment, $n = 10$. Each crawling node runs four processes, with each process maintaining 40 threads. Thus with $n = 10$, the KaZaA Crawling Platform has 1,600 parallel threads. Each thread partially emulates the client-side of the KaZaA connect and query protocol. (We used the results of an earlier reverse engineering project to design the syntax and semantics of the threads' messages [12].) All of these Linux boxes are located in Polytechnic campus in Brooklyn. It is also possible to run crawler experiments from multiple locations distributed throughout the world. However as our measurement results described in section III-C show, we are crawling the vast majority of SNs from the one location itself. Thus a distributed approach is not necessary.

The crawling takes place in rounds of 30 seconds. In each round, each crawling thread operates as follows:

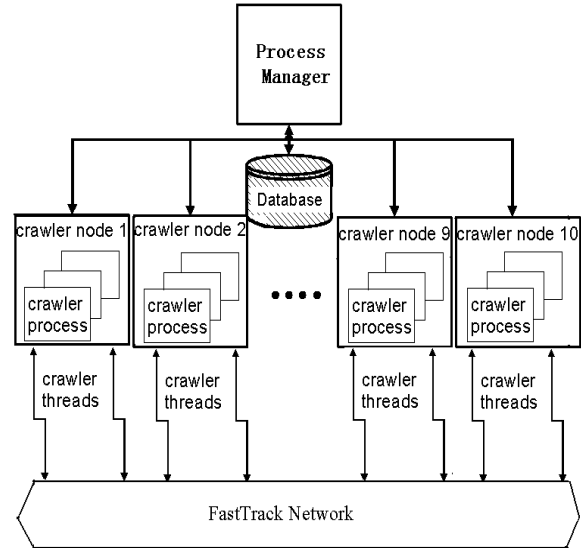1) The crawling thread is initialized with (i) the IP address of some candidate SN in the KaZaA

Fig. 2. *KaZaA Crawling Platform Architecture*

(FastTrack) network, and (ii) a set of query strings. For a targeted song, each query string typically consists of the song title and artist name.

2) The crawling thread attempts to make a TCP connection with the candidate SN. If it fails to establish a TCP connection, then the thread waits until the next round to get a new IP address. If it succeeds, it exchanges handshake messages with the SN and continues as follows.

3) The crawling thread receives from the SN a SN refresh list, consisting of IP addresses of up to 200 SNs. This SN refresh list is forwarded to the Process Manager.

4) For each query string, the crawling thread sends a query to the KaZaA network (via the connected SN). If there are m songs to be queried, the crawling thread sends out m queries back-to-back.

5) For each of these queries, the crawling thread receives (via the connected SN) matching query results. Each query result includes the metadata and ContentHash for the file associated with the match. We set the time-out of each such query session to be 30 seconds.

6) The metadata, ContentHash and IP address from each query result is forwarded to the measurement database.

The Process Manager coordinates and controls the crawling nodes. It maintains a list of all candidate SNs, which is augmented whenever it receives a SN refresh list. In steady state, the Process Manager dispatches 1600 candidate IP addresses to the processes every 30 seconds. Each candidate SN is eventually checked by one of the threads; if the thread succeeds at making a TCP connection with the candidate SN and at querying the SN's local index, the candidate SN is further labeled as *confirmed*.
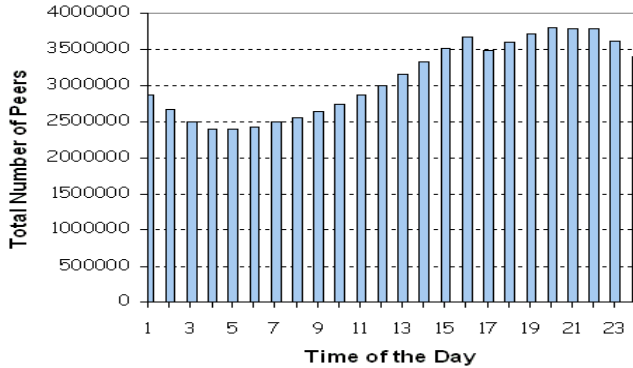
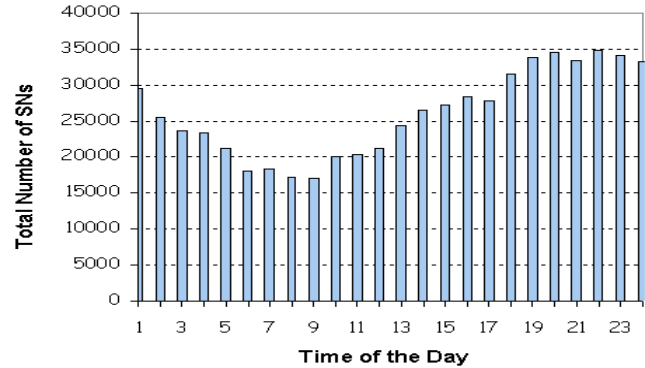At the end of each hour, the KaZaA Crawling Plat-
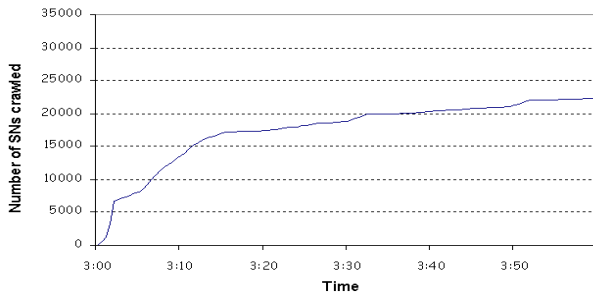
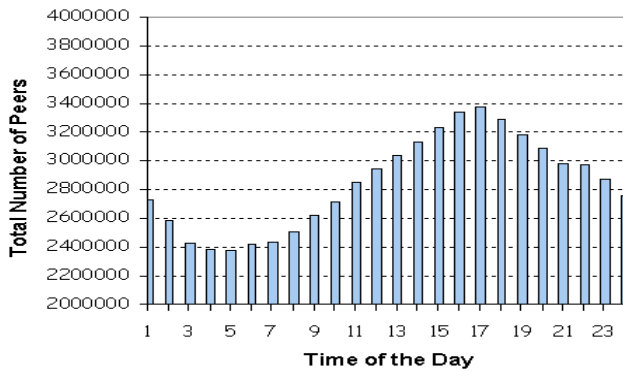(a) May 1,2004 - Early Morning
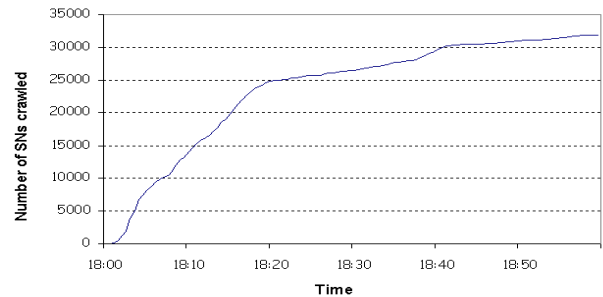
(b) May 1,2004 - Afternoon

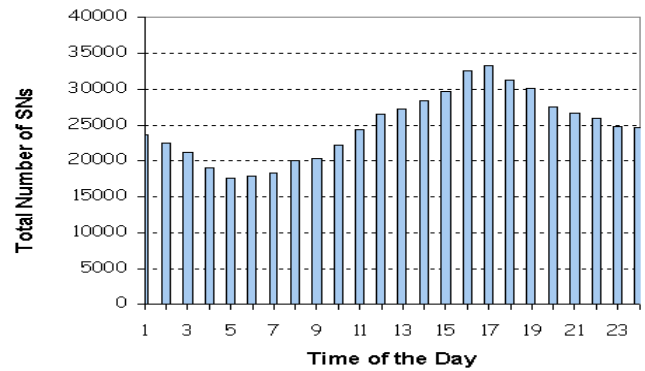Fig. 3. *Number of discovered SNs over one hour*

(a) Total size of the network (SNs + ONs), Saturday, May 1, 2004 (weekend).

(b) Total number of SNs discovered, Saturday, May 1, 2004. (weekend)

(c) Total size of the network (SNs + ONs), Thursday, May 13, 2004. (weekday)

(d) Total number of SNs discovered, Thursday, May 13, 2004 (weekday).

Fig. 4. *Number of SNs and total number of nodes found every hour.*

form starts from scratch, with the candidate set of SNs initialized with the confirmed set of SNs of the previous hour. For each experiment, we gather data for a 24-hour period.

We employ a simple optimization to accelerate the harvest rate of SN IP addresses. As discussed above, after connecting to a SN, a crawling thread sends a sequence of queries into the KaZaA network. We include in this sequence generic queries such as "mp3". For each response, the crawling system identifies the SN that originating the query response. The responses thus provide an additional source of IP addresses, which are merged with the addresses currently in the global list of the process manager. The measurement database contains the metadata, ContentHashes and IP addresses for the song titles under investigation. To protect the privacy of KaZaA users we replace their ON IP addresses with MD5 hashes. We say that two files, stored in different KaZaA files, are **copies** and belong to the same **version** if they have the same ContentHash value. Once the crawling is complete, we perform an offline analysis of the data collected in the measurement database.

C. Crawling Coverage

Recall that in each hour, the crawler attempts to visit as many SNs as possible; and at the beginning of each new hour, the crawling restarts. We claim that in any given hour, the crawler covers the vast majority of SNs that were present in the overlay at sometime during the hour. (Because SNs come and go, the crawler may miss a small fraction of the SNs that were present during the hour. The average lifetime of a SN is about 2.5 hours [12].)

We use two distinct measurement studies to justify this claim. In [12], we determined the number of clients that are connected to a typical SN; we also recorded the total number of peers in KaZaA at any given time, which is provided through the KMD. Dividing the total number of peers by the number of peers connected to a SN gives an estimate of the total number of SNs. We estimated that the number of SNs is about 20,000-30,000, depending on the time of day. Figure 3 presents the number of SNs confirmed by the crawler as a function of time for 60 minutes for two different trials - one in the early morning and the other in afternoon, on the same day. The curves in Figure 3 flatten out after about 30 minutes, at a level of 20,000-30,000 supernodes. The curves do not completely flatten out, however, due to supernode churn. This flattening out of the curves in the 20,000-30,000 range supports our claim that our crawler covers essentially all the supernodes in an hour.

To further justify this claim, we also measured the number ONs and discovered SNs in the overlay for each hour for 24 consecutive hours. The number of discovered SNs in each hour is obtained from the KaZaA Crawling Platform,. For the total number of peers (SNs plus ONs), we again rely on KaZaA's network statistics message displayed in any KMD. Figure 4(a) and 4(c) shows the evolution for the total number of peers in KaZaA while Figure 4(b) and 4(d) shows the evolution of number of discovered SNs in each hour. The measurements were made for over a period of forty eight hours – 0:00 EST May 13, 2004 to 23:59 EST May 1, 2004 for Figure 4(a) and 4(b); 0:00 EST, May 13, 2004 to 23:59 EST May 13, 2004 for Figure 4(c) and 4(d). Observe that the the shape of the evolution of the SNs closely resembles the shape of the evolution of the total number of peers. The average degree of connectivity from a SN to ONs is 115 with a standard deviation of 9.75. The low variance in the degree of connectivity further supports our claim that the KaZaA Crawling Platform identifies almost all the SNs that are present in each one-hour period.

IV. VERSION RESULTS

Many users use applications called "rippers" to extract audio media from compact discs and store extracted audio content on hard drives, where they can be transformed by an "encoder" into the MP3 format

[9]. Such "encoder" and "ripper" processes have recently been bundled into "1-Step" software, making duplication and distribution of mpeg audio files even easier [9]. Typically, the software "ripper-encoders" provide many encoding options, including the bit-rate of encoding and the lengths of silence at beginning and end of the song. Thus, users transform the same song from a CD into non-identical MP3 files, each of which hashes to a different value and is thus a different version of the song. Some of the many other factors that create different versions include ripping of songs from different radio stations, DJ mixes, and, most importantly, different metadata keyed in by different users for the same song. All of this results in a plethora of different versions for the same song, each with its own ContentHash.

We performed an extensive version analysis on the seven popular songs shown in Table I. This analysis is presented in Table II. In choosing the seven songs, we chose songs that were ranked highly in the music charts at the time of the experiment; we also sought a diversity of record labels. Otherwise, our choice was random - we did not select and then reject any songs with any *a priori* knowledge of their version or pollution levels. For each of these seven songs, the KaZaA Crawling Platform determined the number of versions of the song available in the KaZaA network and the number of copies available for each version. As shown in Table II, each of these songs has a huge number of versions, ranging from 8,000 to almost 50,000. The number of copies for these songs is also remarkably large, ranging from about 175,000 to about 1.8 million.

For each of these seven songs, we rank ordered its versions from the most popular to least popular version, where here the popularity of a version is defined in terms of number of its copies discovered in the network. Figure 5 shows the cumulative distribution function (CDF) for the number of copies with respect to the rank-ordered version number. We see from Figure 5 that for each of these songs, more than 60% of the copies come from the top 100 versions and more than 75% of the copies come from the top 500 versions. For two of the songs, more that 90% of copies come from the top 500 versions.

We also plotted the corresponding PMF on a log-log scale in in Figure 6. The linearity of the curves indicates that version popularity closely follows a Zipf distribution, that is, for a given song the popularity of a version is give by

$$f(n) = \frac{K}{n^\alpha}$$

where n is the popularity rank of a given version and $f(n)$ is the fraction of copies of that version discovered in the KaZaA network. For each song we compute α factor by fitting the corresponding data on the log-log plot to a straight line by the least squares method. The
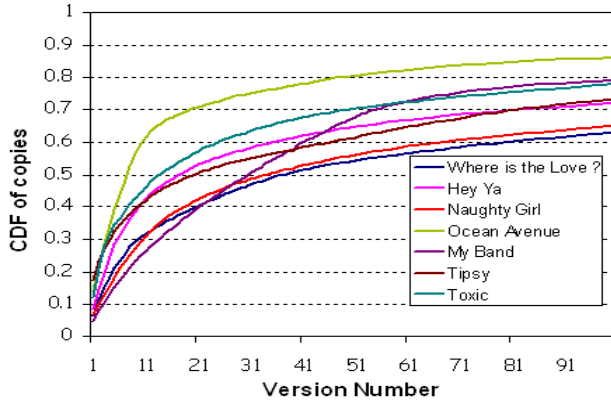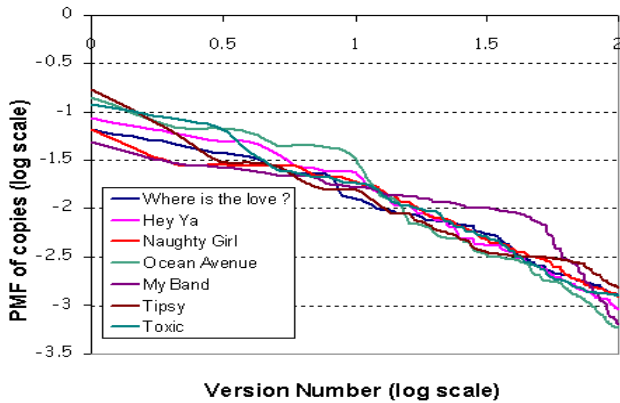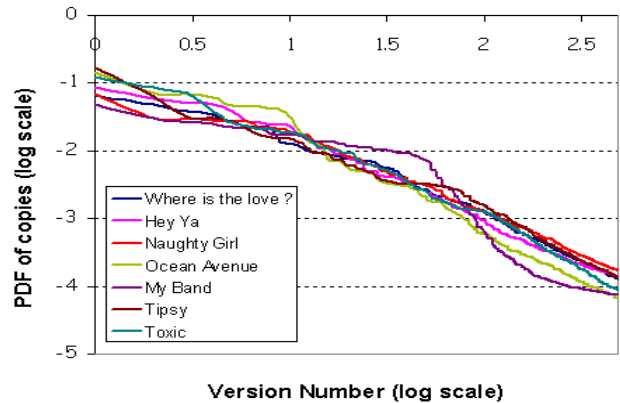
(a) Top 100 versions
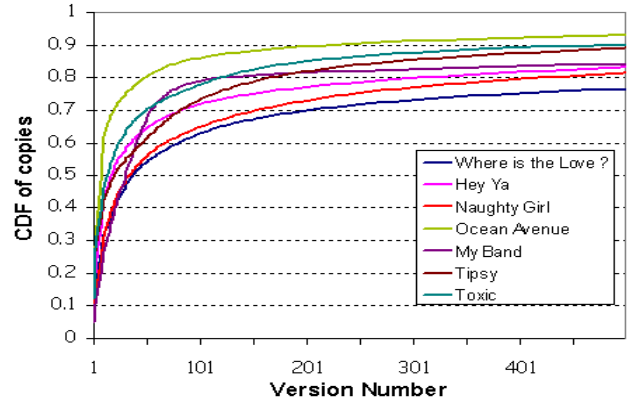
(b) Top 500 versions

Fig. 5. *CDF of copies for 7 songs.*Data collected on May 1, 2004

(a) Top 100 versions

(b) Top 500 versions

Fig. 6. *PMF of copies for 7 songs on a log-log scale.*Data collected on May 1,2004

Song Title	Number of Versions	Number of Copies	α
Naughty Girl	26,715	631,387	0.80672
Ocean Avenue	8,000	174,106	0.80339
Where is the Love ?	48,613	448,987	1.0215
Hey Ya	46,926	734,108	0.86035
Toxic	38,992	650,529	0.86135
Tipsy	32,893	853,688	0.77721
My Band	49,447	1,816,663	0.82019

TABLE II

FOR SEVEN SONGS, NUMBER OF VERSIONS DISCOVERED, NUMBER OF COPIES DISCOVERED, AND ZIPF α VALUE (DATA COLLECTED ON MAY 1,2004)

α factors for the seven songs are between 0.77 and 1.03 and are shown in Table II.

V. POLLUTION

A. Automated Pollution Detection

As described in Section III, the KaZaA Crawling Platform collects the metadata for a keyword string, such as a song title and artist name. For a set of popular songs, we have used the KaZaA Crawling Platform to collect the metadata for essentially all the versions for each of the songs in the song set. Given a particular version for a particular song, the aim of **automated pollution detection** is to determine - without actually listening to the file - whether a version is polluted.

Our detection algorithm is based on the observation that today's polluters use simple techniques for polluting files. In particular, in today's KaZaA network,
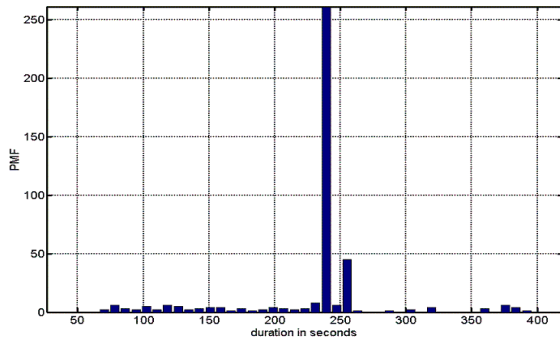
(a) for song Hey Ya
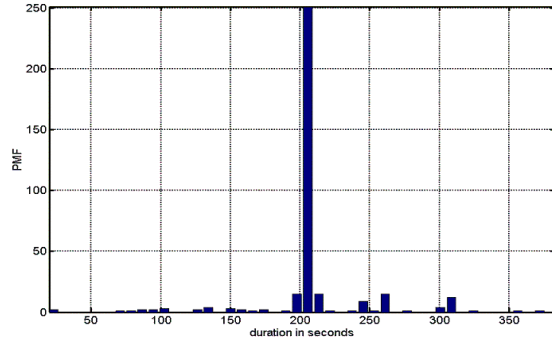
(b) for song Naughty Girl

Fig. 7. *PMF of version durations* shows the PMF for durations for the decodable versions for the songs "Hey Ya" and "Naughty Girl". The official CD durations for these songs are 235 seconds and 208 seconds, respectively.

the vast majority of the polluted MP3 files are **non-decodable** into the corresponding PCM format. This is because the polluting parties usually tamper with the binary format of the mpeg data, rendering the file unplayable (non-decodable). (Instead of writing downloaded files to disk, we download to memory, decode on the fly, and then release the memory after decoding.) Also, we observed that some polluted versions are decodable but have durations that are significantly shorter or longer than the official CD version. *Our simple procedure declares a version to be polluted if either the version is non-decodable or if its length was not within +10% or -10% of the CD length.* We call this last criterion the 10% criterion. For two songs, Figure 7 provides the probability mass functions (PMFs) for the durations for the decodable versions. Note the presence of a significant number of polluted too-short and too-long versions for both songs.

Our pollution detection procedure never creates false positives, that is, it never declares a version to be polluted when it truly isn't. However, it is is possible that the procedure declares some files as clean (that is, as non-polluted) when they are actually polluted. This can happen as follows.

- It is possible that the polluting party actually took care to preserve the mpeg structure of the polluted file. Such a polluted file will decode perfectly and thus pass undetected through our pollution detection procedure.
- All meta-data pollution, as described in Section II, will go undetected.

We performed a statistical analysis to estimate the percentage of false negatives in our pollution detection procedure for the two songs "Hey Ya" and "Naughty Girl." For these songs, we put the versions in persistent storage and manually listened to all 239 versions of "Hey Ya" and all the 412 versions of "Naughty Girl" that were declared clean by our pollution detection procedure. For "Hey Ya", we found 4 content-polluted versions and 13 meta-data-polluted versions, giving the fraction 0.07 of false negatives. For Naughty Girl, we found 17 content-polluted versions and 16 meta-data

versions, giving the fraction 0.08 of false negatives. Thus the pollution statistics reported in this paper are representative of the actual pollution levels in KaZaA.

B. Pollution Results

We use two measures for pollution levels for a given song: the fraction of polluted versions, and the fraction of polluted copies. Figure 8 shows both of these measures for seven songs. The x-axis depicts the titles of the songs and on y-axis the fraction of polluted versions and copies. For example, for the song "Naughty Girl" among the top 500 most popular versions, 62% of the versions are polluted and 73% of the copies are polluted.

From Figure 8 we see that recent popular songs have extraordinarily high levels of pollution in KaZaA. Why? Since pollution is high and widespread over a variety of recent popular songs, we can rule out accidental "defective ripping" by the users as responsible for the bulk of the pollution. We therefore conclude that the music industry is succeeding in generating high pollution levels for popular recent songs. It is remarkable that among the top 500 versions for each of the seven songs considered, the number of polluted versions lies in a range of 100-350.

We emphasize that the levels of pollution shown in the Figure 8 are lower bounds of the actual pollution levels in the network. Indeed, the presence of false negatives, which are versions which pass our decodability test for pollution as described in section V-A, biases the results. We estimate that after taking into account false negatives, the percentage of polluted versions will increase by a value in the range of 7% to 8% (this value comes from our estimation of false negatives in section V-A).

It is also interesting to note that the the two songs which respectively have the highest and lowest levels of pollution also have the highest and lowest number of copies. Specifically, "Ocean Avenue" with the least numbers of versions and copies also has the lowest pollution level. On the other hand, "My Band", with the highest number of versions and copies, has the highest pollution levels. This correlation is also present to a
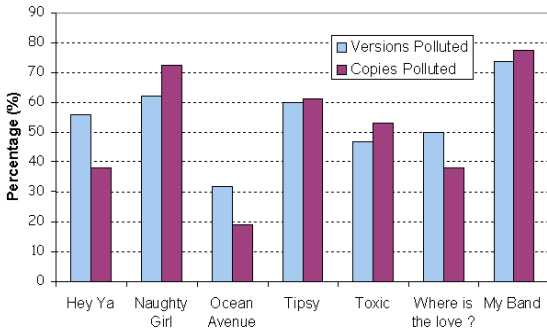
(a) for the top 100 versions
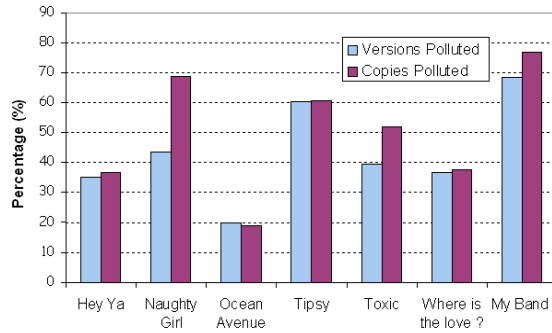
(b) for the top 500 versions

Fig. 8. *Fraction of versions and copies found to be polluted.* Data collected on May 1,2004
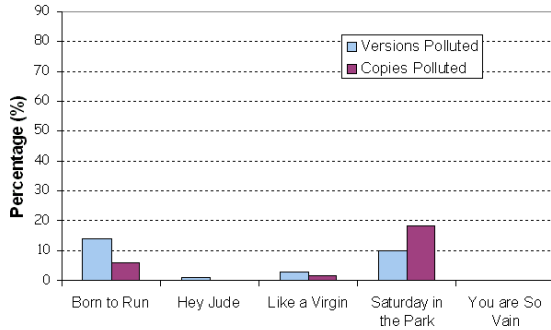

Fig. 9. *Fraction of versions and copies found to be polluted for older songs.* Data collected on June 10, 2004.

large extent in the five other songs. This correlation is likely because the songs with the most versions are the most popular - and hence potentially the most profitable for the music industry. Since the music industry wants to maintain its profits, it more aggressively pollutes the more popular songs.

Also, if an attempt is made to attack a particular song by depositing one or more polluted versions into the network, then it is only worthwhile to do so if the number of copies of these polluted versions is substantial. For this reason, the fraction of polluted copies in the top 100 versions typically exceeds the fraction of polluted versions in the top 500 versions.

To gain insight on what types of songs are highly polluted, we repeated the entire crawling experiment for five older songs (all of which were chart hits in the 70s). These five songs are listed in Table III. From Table III we first observe that these formerly popular songs have relatively few versions and copies, a result which is not unexpected. From Figure 9, we see that the pollution levels for these songs are low, with three of the five songs having less than 2% polluted copies. The pollution levels of "Born to Run" and "Saturday in the Park" are somewhat higher, but still way below those of the currently popular songs. It is possible that most (or even all) of the pollution for these older songs is unintentional pollution.

C. Evolution of Pollution

We also studied the dynamics of content evolution, which, to our knowledge, has not been explored previously. Specifically, we tracked the total number of polluted and unpolluted copies available for the top 100 most popular versions of a given song over a period of 19 days. Due to space constraints, we present the results of this experiment for only two songs, "Hey Ya" and "Naughty Girl". These results are shown in figure 10. We also performed a statistical analysis of this evolution data and found that although the total number of copies available (polluted and unpolluted) is very dynamic, the percentage of polluted copies is slowly varying. The average change in percentage of polluted copies in consecutive measurements is 0.7% for "Hey Ya" and a slightly higher value of 2.5% for "Naughty Girl". This can also be seen by observing that the shape of polluted copies curve closely follows the unpolluted copies curve. It suggests that the dynamics observed in the evolution of content are highly influenced by the change in the size of the network over the experiment duration.

D. Ratings and Pollution

The KMD client gives users the ability to rate the integrity of the files that they are making available for sharing. Any file can be rated as:

- Excellent: File has complete data and is of an excellent technical quality
- Average: File has some of the claimed data and is of moderate technical quality.
- Poor: Poor technical quality.
- Delete File: File may be virus infected or in general should not be shared.

When a user receives responses for a search for a file, the user's KMD client aggregates, for each discovered version, the ratings of all the copies found for that version into one single rating. For example, during a search, if three copies are discovered for some version, and the ratings for the three versions are excellent, poor and null (no rating), the KMD presents to the user the aggregation of these three scores.

SONG TITLE	ARTIST	NUMBER OF VERSIONS	NUMBER OF COPIES
Born to Run	Bruce Springsteen	332	18,828
Hey Jude	Beatles	636	115,846
Like a Virgin	Madonna	326	10,448
Saturday in the Park	Chicago	283	9,331
You're So vain	Carly Simon	261	17,397

TABLE III

OLDER SONGS: NUMBER OF VERSIONS AND COPIES (DATA COLLECTED ON JUNE 10, 2004)
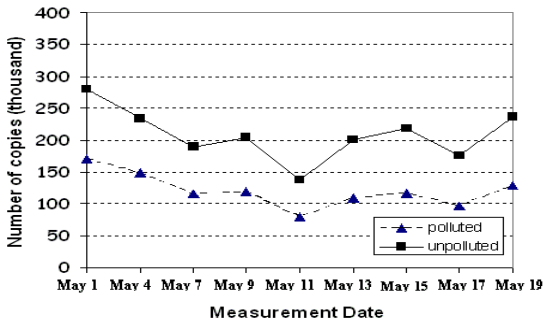
(a) for song Hey Ya
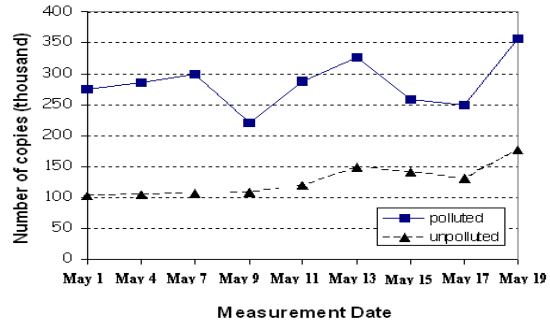
(b) for song Naughty Girl

Fig. 10. *Evolution of Pollution* Shows the number of polluted and unpolluted copies for top 100 most popular versions for the songs "Hey Ya" and "Naughty Girl".

SONG TITLE	% COPIES RATED	P(polluted/good rating)	POLLUTED COPIES(%)
Naughty Girl	1.07	.49	68.9
Ocean Avenue	1.47	.07	19.0
Where is the Love ?	1.83	.04	37.6
Hey Ya	1.82	.21	36.9
Toxic	1.49	.02	51.7
Tipsy	1.61	.17	60.8
My Band	0.80	.52	76.8

TABLE IV

EFFECTIVENESS OF KAZAA'S RATING SYSTEM

For each of the seven recent songs studied in this paper, we recorded the rating for each discovered copy. Table IV provides a summary of our findings. We see from this table that a small percentage of copies are rated for each song. Although the KMD provides incentives for users to rate files by awarding users more participation points whenever a rated file is uploaded [14], the low percentage of rated files is surprising. This is most likely due to (i) the popularity of the KaZaA-lite client, which provides users with maximum participation levels by default, and (ii) lack of user awareness about the relationship between rating activity and participation points.

Table IV also presents statistics on the accuracy of the ratings for the seven songs. We say a copy of a version is **falsely rated** when it has been rated as good (excellent or average) when in-fact it is polluted. The third column of table IV presents the fraction of falsely rated copies for each song. We observe that this fraction is highly correlated with the actual pollution levels, given in Column 4 of the same table: The higher the fraction of falsely rated copies for a song, the higher is the corresponding pollution level. This leads us to conclude that pollution companies also falsely rate their polluted copies.

It appears that even before users are able to rate out a polluted version, new polluted versions are introduced into the network. Frequently introducing polluted versions of a particular song is capable of defeating the content rating mechanism. KaZaA's content rating mechanism is meaningless in the face of an onslaught of polluted versions.

VI. ANTI-POLLUTION MECHANISMS

Given that pollution in P2P file sharing systems is pervasive, it is natural to consider what can be done to defend against the pollution attack. In this section we describe a number of potential anti-pollution mechanisms. We classify the mechanisms into two categories:

- **Detection without downloading:** After receiving search results, the mechanism attempts to determine whether the files in the results are polluted without actually downloading any portion of the files.
- **Detection with downloading:** For this class, the mechanism detects whether a file is polluted by first downloading portions (or all) of the file.

Clearly, from the perspectives of the user and of network traffic, the first class of mechanisms is preferable, as resources are not wasted downloading high-bit-rate polluted multimedia files (or portions thereof).

Detection with file downloading

Within this class of anti-pollution mechanisms, we have identified a number of subclasses:

- **Matching:** In a matching mechanism, there is a trusted database (centralized or decentralized) that contains the fingerprints of authentic content. A fingerprint could be the hash of a song, a frequency (Fourier) representation of the song, or a time-domain summary of the song. For matching, after the client peer downloads a portion (or all) of the file, it matches what it has downloaded with the fingerprints in the database. If a match is not found, the client peer concludes that the file is polluted and deletes all file portions it has downloaded. The Sig2dat project [15] makes available a tool for obtaining the KaZaA ContentHash of any file. This tool is increasingly being used by KaZaA users, who post file names and corresponding ContentHash values on Web sites and message boards. However, because users can easily create different versions of clean (non-polluted) songs, it is unlikely that a hash-based fingerprinting scheme will be successful. Audible Magic [16] offers a proprietary database of frequency-representation of signatures of copyrighted audio content. The database can be used to verify if a file distributed by a P2P file sharing system is copyright protected or not. However, we are skeptical about frequency fingerprinting schemes, as we believe that they can be circumvented by clever content pollution mechanisms. It remains an open question whether there is a time-domain fingerprinting scheme that is difficult to thwart. In any case, each of the fingerprinting schemes requires a trusted database, which not only has a maintenance cost but could itself be the target of a legal attack.
- **User filtering:** We conjecture that if most users first check their downloaded files before copying them into their shared folders, then the level of pollution in file sharing would be significantly reduced. Peers with such a behavior would be acting as sieves, downloading both polluted and unpolluted content but filtering out the former. The challenge here is to provide users a robust incentive scheme that encourages users to filter out polluted files.

Detection without file downloading

The mechanisms in this class rely on the experience of other peers with shared files. For these mechanisms, although a given peer does not need to explicitly download the content, the success of the mechanism depends on an appraisal of the content by other peers that have downloaded the content in the past.

- **Rigid trust:** In this scheme, a user only downloads files from friends who the user fully trusts. These friends agree to manually (listen or watch) verify that files are clean before copying them into their shared folders. If a user starts to receive polluted files from any friend, the user ceases to download files from that friend. Users may locate their friends using presence detection (as in instant messaging systems).
- **Web of trust:** Here the user receives updated lists of friends from all of its own trusted friends. The user downloads from friends and from the friends of his friends. If the user receives polluted content from any friend of friend, the user ceases to download from the friend of friend and notifies his direct friend of the problem. The idea is similar to the trust mechanism used in PGP [17].
- **Reputation systems:** Reputation systems such as [18], [19] allow peers to rank each other. These reputation systems can potentially be used to reduce pollution. The reputation system would identify malicious peers that have been responsible for injecting polluted content into the file sharing system. In an ideal reputation system, peers engaging in malicious behavior eventually develop low reputations. However, current designs of these reputation systems may suffer from problems of low robustness against collusion, and high complexity of implementation.

VII. RELATED WORK

There are a number of other P2P measurement studies, but most of these studies examine *transmitted* P2P traffic rather than *stored* P2P content (as in this paper). In these traffic studies, traffic is collected at a link interface (for example at the boundary of a campus network) and then processed off-line. [20] talks about P2P application specific signatures; these signature techniques could be deployed by an ISP to identify and filter illicit P2P traffic. [21] analyzes P2P traffic by measuring flow-level information collected at multiple border routers across a large ISP-network. By measuring KaZaA traffic

in the University of Washington campus, [1] studies file-sharing workloads and develops models for multimedia workload.

A crawling system was previously developed for the Gnutella P2P network [11]. Developing a crawling system for KaZaA is significantly more challenging for two reasons. First, KaZaA is 10-100 times larger than Gnutella, both in terms of the number of peers and traffic. Second, and more importantly, the Gnutella protocol is in the public domain, whereas the KaZaA protocol is proprietary with little information available to the research community about how it operates. See also [22] for some additional work on crawling Gnutella and Napster.

There has been some recent measurement work on spread of spyware in networked systems. In [23] the authors develop signatures for popular spyware and obtain traces of network activity within the University of Washington campus to quantify the spreading of these programs.

VIII. CONCLUSION

We examined the nature and extent of pollution in P2P file sharing. We found that popular contemporary songs can have a remarkably large number of different versions, as many as 50,000. There are also huge numbers of copies of popular songs, often over 1 million. We found that pollution is indeed pervasive in file sharing, with more than 50% of the copies of many popular recent songs being polluted in KaZaA today. Our results indicate that the vast majority of this pollution is intentional. For older songs, pollution is less prevalent and may mostly consist of unintentional pollution. We have also tracked the evolution of copies in KaZaA and have found that pollution levels remained roughly constant over a 19-day period. We also found that KaZaA's rating system is largely ineffective at identifying polluted copies. We identified and reviewed a number of potential anti-pollution mechanisms.

We developed the KaZaA Crawling Platform to obtain measurement data for this study. This crawler is of independent interest. Developing the crawler was challenging since KaZaA uses a proprietary protocol with most of its signaling messages being encrypted. Also, a farm of server nodes, each running a large number of threads, was necessary to crawl the 20,000+ KaZaA supernodes in an acceptable amount of time. In future work we will further exploit the crawler to gain insight into IP and geographic information on the sources of content.

REFERENCES

[1] 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,*" Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), October 2003.

[2] "*From Discs to Downloads,*" Forrester Research, Inc. http://www.forrester.com/Info/0,1503,353,00.html

[3] "*Americans Continue to Embrace Potential of Digital Music,*" Tempo: Researching the Digital Landscape,http://www.ipsos-na.com/dsp_tempo.cfm.

[4] F. Oberholzer, K. Strumpf, "*P2P's Impact on Recorded Music Sales,*" Second Workshop on Economics of Peer-to-Peer Systems, Cambridge, Massachusetts, June 2004

[5] J. Kurose, K.W. Ross, "*Computer Networking: A Top-Down Approach Featuring the Internet,*" Addison-Wesley, 2005.

[6] "*Overpeer,*" http://www.overpeer.com

[7] "*Hitting P2P Users Where It Hurts,* " Wired News, Jan 13,2003, http://www.wired.com/news/digiwood/0,1412,57112,00.html

[8] "*Method of preventing reduction of sales amount of records due to digital music file illegally distributed through communication network,*" US PATENT AND TRADEMARK OFFICE, June 27, 2002.

[9] "*Method to inhibit the identification and retrieval of proprietary media via automated search engines utilized in association with computer compatible communications network,*" US PATENT AND TRADEMARK OFFICE, May 4, 2004.

[10] "*RetSnap,*" http://www.retsnap.info/

[11] 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, vol. 6, no. 1, 2002.

[12] J. Liang, R. Kumar and K.W. Ross, "*Understanding KaZaA,*" submitted, 2004

[13] "*KaZaA Lite 2.10,*" http://www.k-lite.tk/

[14] "*Integrity Rating,*" http://www.kazaa.com/us/help/glossary/ratings.htm

[15] "*Sig2dat tool for FastTrack network,*" http://www.geocities.com/vlaibb/tools.html

[16] "*Audible Magic,*" http://www.audiblemagic.com

[17] P. Zimmerman, "*Pgp:Source Code and Internals,*" MIT Press, 1995.

[18] S.D. Kamvar, M.T. Schlosser and H. Garcia-Molina, "*The EigenTrust Algorithm for Reputation Management in P2P Networks,*" Proceedings International WWW Conference, Budapest, Hungary, 2003.

[19] D. Dutta, A. Goel, R. Govindan, and H. Zhang, "*The Design of a Distributed Rating Scheme for Peer-to-Peer Systems,*" Workshop on Economics of Peer-to-Peer Systems, June 2003.

[20] S. Sen, O. Spatcheck and D. Wang, "*Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures,*" Proceedings International WWW Conference, New York, USA.

[21] S. Sen and J. Wang, "*Analyzing Peer-to-Peer Traffic Across Large Networks,*" ACM/IEEE Transactions on Networking, Vol. 12, No. 2, April 2004.

[22] K.P. Gummadi, S. Saroiu and S.D. Gribble, "*A Measurement Study of Peer-to-Peer File Sharing Systems,*" Proceedings of Multimedia Computing and Networking, January 2002 (MMCN'02), San Jose, CA, USA.

[23] S. Saroiu, S.D. Gribble and Henry M. Levy, "*Measurement and Analysis of Spyware in a University Environment,*" Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, California, March 2004.