Blacklisting Polluters and Determining Polluted Content in P2P File Sharing Systems

Submission Number 246

***Abstract*— Many P2P file sharing systems are subject to the pollution attack, whereby corrupted copies of popular titles are aggressively introduced into the system. In this paper we explore defenses to the pollution attack. Specifically, we develop efficient mechanisms to determine (i) the IP ranges responsible for introducing polluted content (that is, the blacklist set); and (ii) the pollution levels of titles (songs, movies, etc.) that have been targeted for attack. The methodology is efficient in that it does not involve downloading files from file-sharing nodes. Instead it determines the blacklist set and the targeted content by crawling the file-sharing system, harvesting metadata, and analyzing the metadata. We illustrate the technique by harvesting metadata from the FastTrack/Kazaa network. We apply our methodology to the harvested metadata to determine a blacklist set and the pollution levels of investigated titles. Analyzing the false positives and false negatives we conclude that the methodology is efficient and accurate.**

I. Introduction

By many measures, P2P file sharing is the most important application in the Internet today. There are more than 8 million concurrent users that are connected to either FastTrack/Kazaa, eDonkey and eMule. These users share terabytes of content. In the days of Napster (circa 2000), most of the shared files were MP3 files. Today the content includes MP3 songs, entire albums, television shows, entire movies, documents, images, software, and games. P2P traffic accounts for more than 60% of tier-1 ISP traffic in the USA and more than 80% of tier-1 traffic in Asia [1].

Because of the their decentralized and non-authenticated nature, P2P file sharing systems are highly susceptible to "pollution attacks". In a pollution attack, the "polluter" first tampers with targeted content, rendering the content unusable. It then deposits the tampered content in large volumes in the P2P file sharing system. Unable to distinguish polluted files from unpolluted files, unsuspecting users download the files into their own file-sharing folders, from which other users may then later download the polluted files. In this manner, the polluted copies of a title spread through the file-sharing system, and the number copies of the polluted title may eventually exceed the number of clean copies. The goal of the polluter is to trick users into repeatedly downloading polluted copies of the targeted title; users may then become frustrated and abandon trying to obtain the title from the file-sharing system. The polluter may work on the behalf of a copyright holder, or simply may be a malicious user who wants to prevent an artist from distributing its titles over the file-sharing system. Pollution is currently highly prevalent in file-sharing systems, with as many as 50% to 80% of the copies of popular titles being polluted [2].

In this paper we study mechanisms to limit the effectiveness of the pollution attack. We emphasize, however, that we do not take a side in the P2P file-sharing debate, neither condoning nor condemning the pollution attacks that are commissioned by the music, television and film industries. But given that P2P file sharing traffic is currently the dominant traffic type in the Internet, and that the files being transferred are frequently polluted, a significant fraction of Internet bandwidth is clearly being wasted by transporting large, corrupted files. Furthermore, artists who want to distribute their content via P2P file sharing systems will always be susceptible to pollution attacks. It is therefore important to gain a deep understanding of the pollution attack and develop effective mechanisms to counter it.

In this paper we explore two techniques for countering the pollution attack:

- **Blacklisting IP address ranges:** The goal is to identify IP address ranges that are broad and complete enough to cover the polluters' hosts yet narrow enough to exclude the vast majority of ordinary users.
- **Identifying the targeted titles:** With knowledge of which titles are under attack, ordinary users can more intelligently make decisions about what to download.

In developing methodologies for these two counter-pollution techniques, we have not only aimed for accuracy but also for efficiency. For blacklisting, one approach would be to download copies of titles from a vast number of IP addresses and then manually check the copies for pollution; the IP addresses that consistently supply polluted content could then be blacklisted. Such an approach would be highly inefficient, requiring enormous bandwidth, computing and human resources, and would also introduce significant "probing" traffic into the Internet.

Our methodologies do not involve the downloading of any files. Instead, they identify polluting IP address ranges and targeted titles by collecting and analyzing metadata from the file sharing system. The metadata is harvested by crawling the nodes in the P2P system and sending tailored queries to each of the crawled nodes. The harvested metadata can then be analyzed to obtain detailed information about the numbers of versions and copies, and the IP subnets containing the versions and copies, for a large number of investigated titles. From this detailed information, our methodology constructs the blacklisted IP ranges and the estimated pollution levels for the targeted titles. The methodology is efficient in that it collects metadata (text) rather than content (which is typically

3MB to several GB per file for music and video) and that a large number of titles and virtually all the file-sharing nodes can be investigated in one crawl.

Our contribution is as follows:

- We developed a methodology for creating a blacklist set. The methodology is based on identifying high-density prefixes, which are prefixes in which the nodes that have a copy of a particular title have, on average, a large number of copies. We provide a heuristic for separating the low density prefixes from the high density prefixes, and a mechanism to merge prefixes that are topologically close. The set of resulting merged prefixes constitutes the blacklist set. We then developed several metrics for measuring the accuracy of the blacklist set. The two principal metrics are probability of false positive and false negative. We also examine secondary metrics, including comparing the download times and last-hop RTTs at blacklisted and non-blacklisted nodes.
- We developed a methodology for estimating the pollution level of a title, which is defined as the ratio of polluted copies in the network to the total number of copies in the network. This estimate does not involve the downloading of any files and is solely based on the harvested metadata. We then evaluate our estimate by measuring the actual pollution levels of selected titles.
- We crawled FastTrack for 170 titles, including songs and movies. We then applied the methodologies to the FastTrack metadata harvested during the crawling procedure. Our resulting blacklist set contains 112 prefixes. Our evaluation metrics indicate that the blacklist set is accurate, with low probabilities of false positives and false negatives. We also find that the estimates for pollution levels in examined titles is accurate.

This paper is organized as follows. In Section II we describe the pollution attack in detail and introduce important terminology. In Section III we describe in detail the methodologies and the evaluation procedures for creating the blacklist set and the pollution-level estimates. In Section IV we describe the experimental setup, including the crawler and PlanetLab experiments. Section V provides the results of our experiments, including evaluation results for the methodologies. Section VI describes previous work related to this paper. We conclude in Section VII.

II. Overview of Pollution

A. File Sharing Terminology

We first provide an overview of a generic P2P file-sharing application. This will allow us to introduce some important terminology that is used throughout the paper. In this paper we are primarily concerned with the sharing of music and video. We shall refer to a specific song or video as a **title**. Examples are titles include "studio recording of Beatles' Something," "Seinfeld Episode 17," and "Matrix". When a title is shared in a P2P file sharing system, it is typically compressed in some compression format (such as mp3, rm, wma, mpg, avi, mov) at some compression rate and then introduced as a file. Importantly, a given title can have many different **versions** (in fact, tens of thousands). These versions primarily result from a large number of rippers/compressors, each of which can produce slightly different files when created by different users. Modifications of metadata can also create different versions. Users download different versions of titles from each other, thereby creating multiple **copies** of identical file versions in the P2P file sharing system. At any given time, a P2P file-sharing system may make available thousands of copies of the same version of a particular title.

A file in a P2P file sharing system typically has **metadata** associated with it. There are two types of metadata: metadata that is actually included in in file itself and is often created during the ripping process (e.g. ID3 Tags in mp3 files); and metadata that is stored in the file-sharing system but not within the shared files themselves. This "outside-file" metadata may initially be derived from the "inside-file" metadata, but is often modified by the users of the file-sharing systems. It is the outside-file metadata that is employed during P2P searches. In this paper, when using the term metadata, we are referring to the outside-file metadata. Because different copies of a version of a title may be stored on different user nodes, the different copies can actually have different metadata.

When a user wants to obtain a copy of a specific title, the user performs a keyword search, using keywords that relate to the title (for example, artist name and song title). The keywords are sent within a query into the file-sharing network. The query will visit one or more nodes in the file sharing network, and these nodes will respond if they know of files with metadata that match the keywords. The response will include the metadata for the file, the IP address of the node that is sharing the file, and the username at that node. For many file sharing systems, the response will also include a **hash**, which is taken over the entire version. To download a copy of a version, one sends a request message (often within an HTTP request message) to the sharing user. In this request message, the version is identified by its hash. Many file sharing systems employ parallel downloading, in which case requests for different portions of the version are sent to different users sharing that file.

Many nodes in P2P file sharing systems are behind Network Address Translators (NATs). If a node is behind a NAT, it can download files; it can also upload files to non-NATed nodes using a technique known as a "reverse connection". Non-NATed nodes have **public IP addresses**, and NATed nodes **private IP addresses**; the NAT itself has a public IP address. When crawling a P2P file sharing network, for a NATed node sharing relevant files, the crawler may return a node's private IP address and private port number rather than its public IP address and public port number. Since the range of private IP address is relatively narrow, different NATed users may have the same private IP address. Thus, from the crawling data, we cannot distinguish between different users solely by their IP addresses. In order to distinguish between different users, including NATed users, we define a **user** as the triple

(IP address, port number, username).

B. *The Pollution Attack*

The pollution attack is initiated by a **polluter**. The polluter may work on the behalf of a copyright holder; or the polluter may be a malicious user who wants to prevent an artist from distributing its titles over the file-sharing system. The polluter typically takes the following steps when attacking a specific title:

1) The polluter creates one or more polluted versions of the title. This is done by tampering with 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, and inserting advertisements. We have observed that today a popular pollution technique is to insert tens of seconds of undecodable white noise into the middle of the song.
2) The polluter connects one or more nodes to the P2P file-sharing system and places the tampered versions into its shared folders on these nodes.
3) Users query for the title and learn about the locations of versions of the title, including the polluted versions. The query results provide no indication of which versions are clean and which are polluted.
4) Some users download polluted versions. After a user downloads a polluted version, it may remain in the user's shared folder for an extended period of time. This is because the user may not immediately watch or listen to the file and detect that it is polluted; or the user may simply neglect to remove the file even after observing that it is polluted.
5) The polluted version spreads through the file-sharing system. Unsuspecting users continue to download the polluted version either from a polluter or from an ordinary user.

Pollution is prevalent in modern P2P file sharing systems such as FastTrack/Kazaa [2]

Most of the pollution today emanates from "professional" polluters that work on the behalf of copyright owners, including the record labels and the motion-picture companies. From this economic context and from our own testing and usage experience, we conclude that the professional polluters have the following characteristics.

- Polluters tend to pollute popular content, such as recently-released hit-songs and films. Indeed, the popular content is the most lucrative content for the copyright owners. The copyright owners commission companies to spread polluted versions of their popular content throughout the P2P networks, thereby curtailing the free distribution of the content. In [2] a random sample of recent, popular songs were shown to be heavily polluted whereas a random sample of songs for the 70s were shown to be mostly clean.
- Polluters have high-bandwidth Internet connections. The polluters can thus upload content at least as fast as ordinary users, which typically have residential broadband or campus connections. The polluters essentially create honeypots by offering content at attractive bandwidths.
- Polluters have high availability, that is, the polluters' nodes provide stable and continuous upload service for long periods of time.
- Polluters are not behind firewalls or NATed routers. If a polluter is behind a NAT, then it would be more difficult for a user to download content from it, particularly if the user is also behind a NAT. Because polluters want users to download content from them, they naturally locate their nodes in front of firewalls and NATs.

In our methodology for detecting polluted content and blacklisting polluters, we will make the following additional assumptions about polluters. Many of these assumptions will be corroborated in Section V, where our measurement results are presented.

- Because polluters share popular titles at attractive file-transfer rates, there is a high demand for their content from unsuspecting users. To meet the demand, the polluter often uses a server farm at one or more polluter sites. The nodes in a server farm are concentrated in a narrow IP address range.
- Whereas regular P2P users run one client instance per host, polluters often run many clients in each of their nodes, with each instance having a different username and sharing its own set of copies and versions for the targeted titles. This is done to improve placement of search results in the users' GUIs.
- A polluter distributes multiple polluted versions of the same title. This also improves the placement of search result in the users' GUIs. As we will show in Section V, an ordinary user typically has a small number of versions of any title. To compete with all the clean versions in the display of the search results, a polluter needs to provide many different versions (each with a different hash) to increase the chances that its versions are selected from the users' GUIs.

C. *Pollution Evolution*

Our observations of pollution in P2P networks has led us to define the following three stages that a title can be in during the pollution attack.

Introductory Stage: When a title is released and is targeted by the pollution attack, the polluter will introduce many versions with many copies per version of the title into the file sharing system. (The polluter may actually do this before the official release date.) At this stage, the number of polluting users is more than the number of ordinary users, and almost all versions of the title are polluted.

Growth Stage: In the weeks following the release of the title, ordinary users introduce clean versions of the title into the file sharing system. The number of versions and copies grows, as ordinary users download clean and polluted versions and introduce new clean versions. In this stage, the number of ordinary users is larger than the number of polluting users.

Post-Pollution Stage: At some point, the polluters stop targeting the title (and move on to newer titles). The fraction of copies that are polluted declines, and the ratio of ordinary users to polluting users increases. No single user provides significant number of copies or versions of the title.

We use these stages in Section V to classify content with different pollution characteristics.

III. Methodology

In this paper we develop methodologies for two tasks. The first task, which we refer to as **blacklisting**, is to find the IP address ranges that include the large majority of the polluters. The second task, referred to as **pollution level estimation**, is to determine the extent of pollution for specified titles. For both of these tasks, the first step is to crawl the file sharing system, as we now discuss.

A. Crawling

Crawling a P2P file sharing system is the process of visiting a large number of nodes to gather information about the copies of files being shared in the system. The crawler might gather, for example, the IP addresses and hashes of all copies of files being shared in the network for a set of specific titles over a given period of time. Several independent research groups have developed crawlers for P2P file sharing systems. A crawler for the original single-tier Gnutella system is described in [3]. A crawler for the current two-tier Gnutella system (with "ultrapeers") is described in [4]. A crawler for eDonkey is described in [15]. A crawler for the FastTrack P2P file sharing is described in [2]. Since P2P networks are dynamic, with nodes frequently joining and leaving, a good crawler needs to rapidly crawl the entire network to obtain an accurate snapshot.

The first step in our methodologies is to crawl the P2P file sharing system and obtain the following information for each title of interest: the number of versions in the file sharing system for the title; the hash values for each of the versions; the number of copies of each version available in the file sharing system; for each copy, the IP address of the node that is sharing it; the port number of the application instance at that node (many modern P2P systems vary the port number across nodes to bypass firewalls); the username at that node; and, for each copy, some copy details (e.g., playtime, file size, description, etc). For each title of interest, the crawler deposits this information in a **crawling database**, which can then be analyzed off-line. We will describe a crawler for the FastTrack network in Section IV.

B. Blacklisting

Polluters typically control blocks of IP addresses and can easily move their nodes from one IP address to another within the block. Thus, rather than blacklisting individual IP addresses, we should blacklist ranges or IP addresses that are likely to include the polluters in the near future as well as the present. We use approach similar to the one used in [25]. Our methodology for blacklisting has the following steps:

1) Crawl the P2P file sharing system as described above.
2) From the data in the crawling database, identify the /24 prefixes that are likely operated by polluters.
3) Merge groups of /24 prefixes that are topologically close and don't cross BGP prefixes. The set of merged prefixes becomes our blacklist set.

We now describe the second and third steps in more detail.

The second step is to identify /24 prefixes that are likely operated by polluters. A polluter typically leases from a data center a set of server nodes in a narrow IP address range. Data centers do not normally include ordinary P2P users, which typically access the Internet from residences and universities. A /24 prefix is small enough so that both polluters and ordinary users do not operate from within the same prefix; and it is large enough to cover multiple polluting servers in most subnets. In the third step, we search for larger subnets.

Let N denote the number of titles investigated and T_n denote the nth title. For each title T_n, we determine from the crawling database the /24 prefixes that contain at least one copy of title T_n. Suppose there are $I^{(n)}$ such /24 prefixes; denote the set of these prefixes by $\mathcal{P}^{(n)} = \{p_1^{(n)}, p_2^{(n)}, \ldots, p_{I^{(n)}}^{(n)}\}$.

We now introduce the important concept of the "density of a prefix," which will be used repeatedly in this paper. For each such prefix $p_i^{(n)}$, define $x_i^{(n)}$ to be the number of IP addresses in the prefix with at least one copy of the title and $y_i^{(n)}$ to be the number of copies (included repeated copies across nodes) of the title stored in the prefix. Finally, define the **density** of prefix $p_i^{(n)}$ as $d_i^{(n)} = y_i^{(n)}/x_i^{(n)}$.

From our assumptions about how polluters operate (see Section II), we expect the prefixes with high density values to be operated by polluters and prefixes with low densities to contain only "innocent" users. We consider prefixes with a density higher than a threshold $d_{thresh}^{(n)}$ to be operated by polluters. There are many possible heuristics that can be used to determine this threshold. We now describe a simple heuristic that gives good performance. It is based on the median value of the distinct density values in $\{d_1^{(n)}, d_2^{(n)}, \ldots, d_{I^{(n)}}\}$ denoted by $d_{median}^{(n)}$. Of course, different prefixes have different numbers of users and different densities, so in order to allow for a variance in user behavior we set a threshold to a multiple of the median. Specifically, our heuristic sets the threshold to

$$d_{thresh}^{(n)} = k d_{median}^{(n)} \tag{1}$$

where k is an appropriately chosen scaling factor (see Section V). We say that a prefix $p_i^{(n)}$ is a **polluting prefix** if $d_i^{(n)} \geq d_{thresh}^{(n)}$. Let $\mathcal{Q}$ be the union of all the polluting /24 prefixes over all N titles.

A polluter may actually operate within a network that is larger than a /24 prefix. The third step of our methodology is to create larger prefixes which encompass neighboring /24 prefixes in $\mathcal{Q}$. For this, we merge adjacent prefixes in the IP space. We also merge some non-adjacent prefixes. To this end, we perform a traceroute from each of 20 PlanetLab nodes to one IP address in each of the prefixes in $\mathcal{Q}$. Prefixes which have the same last router become candidates for merging. In doing this we need to account for the possibility that some of the traceroutes passing through the same last router may actually pass through the router via different interfaces (and thus IP addresses) [9]. Suppose there are J groups of prefixes, with each prefix in a group sharing the same last router. (Some groups may contain a single prefix.) Let $\mathcal{G}_j,\ j = 1, \ldots, J$, denote the groups. For each group of prefixes $\mathcal{G}_j$, denote p_j as the longest prefix that covers all the prefixes in $\mathcal{G}_j$. For each such p_j we verify that it does not cross prefixes found in a BGP table. If it does, we decompose p_j back into its original /24 prefixes. Let $\mathcal{P}$ be the resulting set of prefixes. $\mathcal{P}$ is our final blacklist set, and consists of all the p_j's that pass the BGP test and all of the decomposed /24 prefixes as just described.

Note that this methodology for creating a blacklist set does not involve the downloading of content. Indeed, any download-based methodology would require the downloading of an excessively large number of files as well as an automated procedure to determine whether a downloaded file is polluted. Our approach is instead based on the metadata that is gathered by the crawler. This approach is efficient in that crawling a large-scale P2P file-sharing system can be quickly done with modest resources.

C. Evaluation Procedure for Blacklisting

The blacklist set $\mathcal{P}$ may not be completely accurate in that it may not contain all polluting nodes (false negatives) and it may contain some active nodes that are innocent users (false positives). We evaluate a blacklisting methodology by estimating the probability of false positives and false negatives.

To this end, we need a procedure to determine whether a downloaded version of any given title is polluted. This can be done by downloading the version and manually watching or listening to it. Such a manual procedure would require an excessive amount of human resources. Instead we use a simple automated procedure which has been shown to give accurate results [2]. Specifically, we download the version into RAM and declare the version to be clean (unpolluted) if the following three criteria are met:

1) Rehashing the file results in the same hash value as the one that was used to request the file;
2) The title is decodable according to the media format specifications; for example, an mp3 file fully decodes as a valid mp3 file [24].
3) The title's playback duration is within 10% of the one specified in release information for that title.

In any one of the three criteria is violated, we consider the version to be polluted. We refer to this procedure as the **automated version-checking procedure**.

Having described our procedure to determine whether a downloaded version is polluted, we can now state our false-negative and false-positive evaluation procedure. To evaluate the false-negative probability, we randomly select 1,000 users having IPs outside of $\mathcal{P}$ and having a copy of at least one of the investigated titles. For each randomly selected node, we randomly download 5 versions stored at that node. We declare a node to be a **false negative** if all of the following conditions are satisfied: (i) it has at least 5 versions of any one of the investigated titles; (ii) its upload throughput is greater than a given threshold. (In Section IV we describe how we estimate a node's upload throughput); (iii) its Last Hop RTT is less than a threshold (we define Last Hop RTT in Section IV) ; and (iv) all of the randomly selected versions are polluted. Thus a randomly selected node is declared a false negative if that node has the main characteristics of a polluting node. (See Section II.) The false-negative probability is simply the the number of randomly selected nodes declared to be false negatives divided by the total number of randomly selected nodes.

A false positive occurs when an "innocent" non-polluting node is blacklisted as a polluter by our methodology. This can happen when a /24 prefix is labelled a polluting prefix but contains non-polluting users, or when innocent users are added to $\mathcal{P}$ during the merging process. To evaluate the false-positive probability, we randomly select 1,000 nodes in $\mathcal{P}$ containing a copy of at least one of the titles. For each randomly selected node, we randomly download five versions stored at that node. We declare a randomly selected node to be a **false positive** if any of the following criteria are satisfied: (i) its throughput is smaller than the threshold; (ii) its last hop RTT is larger than the threshold; and (iii) at least one of the randomly selected versions is clean. The false-positive probability is simply the the number of randomly selected nodes declared to be false positives divided by the total number of randomly selected nodes.

D. Estimating Content Pollution Levels

In this subsection we provide our methodology for estimating the pollution level of any arbitrary title T_n. The methodology builds on the blacklisting methodology. We define the **pollution level** of a title as the fraction of copies of the title available in the P2P file sharing system that are polluted. The pollution level of a title can be estimated by randomly selecting a large number of copies of the title, downloading each of the copies, and then testing the copies for pollution (either by listening to them or through some automated procedure). This requires an exorbitant amount of resources, particularly if we wish to accurately determine the pollution levels of many titles. We instead estimate the pollution levels of titles directly from the metadata available in the crawling database. To this end, we make the following assumptions:

1) All copies of the title that are stored in a blacklisted node (that is, in a node in $\mathcal{P}$) are polluted.
2) For each node outside of $\mathcal{P}$ with at least one copy of T_n, all copies stored at that node are polluted except for one copy.

With these assumptions, we now derive an expression for $E^{(n)}$, the **estimated pollution level** of title T_n. For a title Recall that $y^{(n)}$ is the total number of copies of the title available in the crawling database. Also let $z^{(n)}$ be the number of nodes outside of the blacklist set $\mathcal{P}$ that have at least one copy of T_n.

The above two assumptions imply that the number of copies of T_n that are polluted is $y^{(n)} - z^{(n)}$; thus, our estimate of the pollution level for title T_n is

$$E^{(n)} = \frac{y^{(n)} - z^{(n)}}{y^{(n)}} \tag{2}$$

E. Evaluation Procedure for Pollution-Level Estimation

$E^{(n)}$ is an estimate of the pollution level of title T_n, derived solely from the metadata in the crawling database. To evaluate the accuracy of this estimate, we compare it with a measured value, which is obtained by actually downloading content. Specifically, for a given title T_n we do the following:

1) We download the most popular versions of the title. The number of versions downloaded is such that the downloaded versions covers at least 80% of all copies of the title in the file-sharing system. For title T_n, let J_n be the number of versions that meet this 80% criterion.
2) For each of these versions, we determine if the version is polluted or not using the automated version checking procedure described in Section III-C. Let $\delta_i^{(n)}$ be equal to 1 if version i is determined polluted and be equal to 0 otherwise.
3) The crawling database provides the number of copies that each version contributes. Let $c_i^{(n)}$ be the number of copies of version i in the database. We calculate the fraction of polluted copies $L^{(n)}$ as

$$L^{(n)} = \frac{\sum_{i=1}^{J_n} c_i^{(n)} \delta_i^{(n)}}{\sum_{i=1}^{J_n} c_i^{(n)}} \tag{3}$$

We then define the error in the pollution-level estimate as:

$$Error = \frac{|E^{(n)} - L^{(n)}|}{L^{(n)}} \tag{4}$$

We present the resulting error and its implications in Section V.

IV. Experimental Setup

We evaluated our methodologies from data collected in the FastTrack file-sharing network. We first briefly describe the FastTrack network and the FastTrack crawler.

A. Overview of FastTrack

With more than two million simultaneous active nodes (in Nov 2004), FastTrack is one of the largest P2P file sharing systems. It has at least an order of magnitude more users than Gnutella. It is used by several FastTrack clients including KaZaA, kazaa-lite, Grokster, and iMesh. It is also known to be the target of the pollution attack. For these reasons, we chose to test our blacklisting and pollution-level methodologies on FastTrack.

Unlike Napster, FastTrack is decentralized and does not maintain an always-on, centralized index for tracking the location of files. FastTrack has two classes of nodes, Ordinary Nodes (ONs) and SuperNodes (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 FastTrack 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 hashes for the files it is sharing. This allows the SN to maintain a local index which includes hashes and file descriptions for all the files its children are sharing along with the corresponding IP addresses of the ONs holding the particular files. 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 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.

Many FastTrack nodes are behind NATs. During a query, when a SN responds with information about a NATed node and a file that the NATed node is sharing, the SN responds with the NATed nodes private IP address (as well as the node’s username and dynamic port number). As indicated in Section II, we differentiate among these NATed users by defining a user to be the triple (IP address, port number, username).

B. The FastTrack Crawler Platform

The methodologies described in Section III require us to crawl the FastTrack network. We developed the FastTrack Crawling Platform, which crawls through virtually all of the 30,000+ FastTrack 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. Developing a crawling system for FastTrack is challenging for two reasons. First, FastTrack is huge, with 10-100 times more nodes and traffic than Gnutella. Second, and more importantly, whereas the Gnutella protocol is in the public domain, the FastTrack protocol is proprietary with little information available to the research community about how it operates.

The FastTrack Crawling Platform is shown in Figure 1. It consists of a process manager, a crawling database, and n crawling nodes each implemented in its own Linux box. In our current deployment, $n = 4$. Each crawling node runs 16 processes, with each process maintaining 40 threads. Thus with $n = 4$, the FastTrack Crawling Platform has 2,560 parallel threads. Each thread partially emulates the client-side of the FastTrack Network connect and query protocol. All of these Linux boxes are located on a university campus in North America. It is also possible to run crawler experiments from multiple locations distributed throughout the world. However, we found that a centralized design was sufficient, as it can crawl all of the FastTrack SNs in a short period of time. 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 FastTrack network, and (ii)

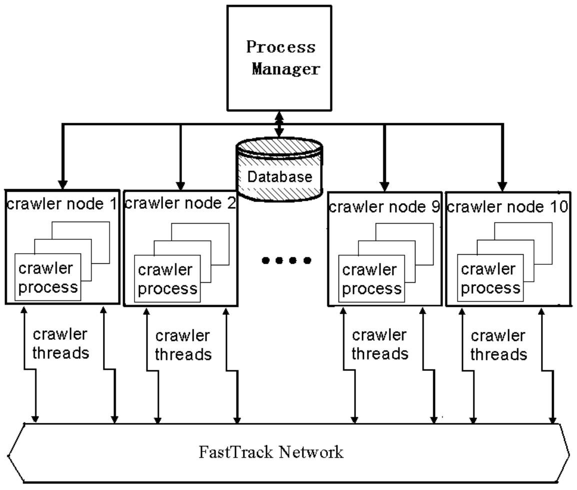


Fig. 1. FastTrack Crawler architecture

a set of query strings. For a targeted title, each query string typically consists of the title name 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 titles 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 hash for the file associated with the match. We set the time-out of each such query session to be 30 seconds.
6) The metadata, hash, IP address, username and port number from each query result is forwarded to the crawling 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 2,560 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 SNs local index, the candidate SN is further labelled as confirmed.

C. PlanetLab Measurement

As part of our evaluation procedures, we also determine the download throughput and the last-hop Round Trip Time (RTT) at various nodes in the FastTrack network. We now describe how we measure those metrics. Download throughput and last-hop RTT depend on the measurement host from which the measurement is being initiated. To reduce the observer bias, we distributed our measuring hosts on a number of PlanetLab nodes.

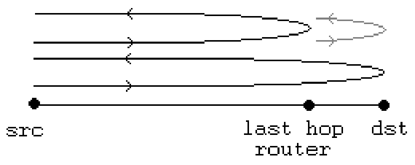


Fig. 2. Computing Estimated Last-Hop RTT: The estimated last-hop RTT (grey) is the difference between the RTT times to the destination host and the last-hop router.

1) Throughput: We used 20 well connected PlanetLab hosts that downloaded data from each measured host. The measurement was performed in the following way. The PlanetLab nodes sequentially establish TCP connections to each measured FastTrack node. For each connection, the PlanetLab host requests 500KB of data. If t denotes the time from when the PlanetLab host begins to receive the 500KB until when it has received all of the 500 KB, then the throughput of the connection is defined to be $(500KB)/t$. The throughput of the node is then obtained by averaging the throughput over all successful connections. A custom program was developed to download and report those measurements from the different nodes.

2) Estimated Last-Hop RTT: To verify our blacklisting methodology, we also measured the Last-Hop Round Trip Time (RTT). The last-hop RTT is the time it takes a small packet to travel from this last router to the destination host and back to the last router [22]. We can only estimate the last-hop RTT with indirect measurement from our sources, since we don't have access to the routers. To estimate the last-hop RTT, we measure the minimum out of 3 RTTs from the source to the destination from which we subtract the minimum RTT from the source to the last router, as shown in Figure 2. For each PlanetLab source, we found the difference of those values and took the minimum one as an estimate of the last-hop RTT for that target IP address.

In both experiments we used 20 PlanetLab nodes and we selected them from different parts of North America, Europe, Asia and the Pacific. Two nodes from each of the domains in Table I were used to produce the total of 20 nodes. We believe that this carefully chosen set is representative for our purposes.

V. RESULTS

In this section we present the results of our experiments. We first describe the raw data. We then provide the results pertaining to the blacklisting and pollution-level methodologies.

TABLE I
PLANETLAB NODES USED FOR DISTRIBUTED MEASUREMENTS

North America	Europe	Asia/Pacific
poly.edu	uni-wuerzburg.de	snu.ac.kr
ucsd.edu	vu.nl	ntu.edu.tw
berkeley.edu	ethz.ch	
cc.gt.atl.ga.us		
utk.edu		

TABLE II
RAW DATA FOR REPRESENTATIVE TITLES

title	versions	users	copies	total IPs	Public IPs
040	225341	135102	2120160	8627	7577
008	155642	112542	1575686	6188	5298
060	91447	172879	300865	57681	52252
052	48607	126226	301075	46129	40419
097	9648	28583	37173	17468	15289
009	3795	41405	56215	24689	22388
111	5503	14519	17562	9801	8437
005	5856	56351	67945	37051	32162

A. Raw Data

The crawling platform captured 124GB of metadata on the KaZaA network from Nov 21 to Nov 27. The crawler queried supernodes around the world for 170 titles, consisting of 133 songs and 37 movies. We choose these title as follows. As discussed in Section 2, popular, newly released titles are often targets for pollution. Even though this paper mostly focuses on music, we include some movie titles to illustrate that the technique is universal. Most of the chosen songs are new popular songs, with their titles obtained from the listing available at itunes.com [6]. Some of the chosen songs are "oldies" obtained from about.com 70s charts [7]. The remaining chosen content is newly released DVDs from the list of top rentals at blockbuster.com [8]. This selection gave us a varied list of popular titles without particular taste or style preference bias in our results.

As we described earlier, we use the combination (IP address, port, username) to identify a unique user. Because of the difference in the popularity of the titles that we investigated, the number of users that possessed copies of a title varied greatly from title to title, with a minimum of 1,801 users for a title and a maximum of 311,135 for a title. Because some users (especially polluters) have more than one copy of a title, the number of copies of a title is larger than the number of users with the title. The number of copies of a title varied from 2,188 to 2,120,160 copies. Our crawling system captured around 1.3 million unique public IP addresses during the seven-day period that we crawled FastTrack. The titles with less than 10,000 copies were not taken into account when creating the blacklist set $\mathcal{P}$ since they do not provide with enough data for meaningful conclusions. The resulting list contained 122 titles.

Table II includes some of the data that we gathered from the crawler for a few of the representative titles. The titles in this table are chosen to represent a diversity of data distributions. The presented data includes the title numbers; the number of versions (hashes) of that title that were observed; the number of users who possess a copy of any version of title; the number of copies of the title; the total number IP addresses that were gathered (Because multiple FastTrack clients can be present on one IP address, one IP address can represent more than one user.); and the number of public IP addresses.

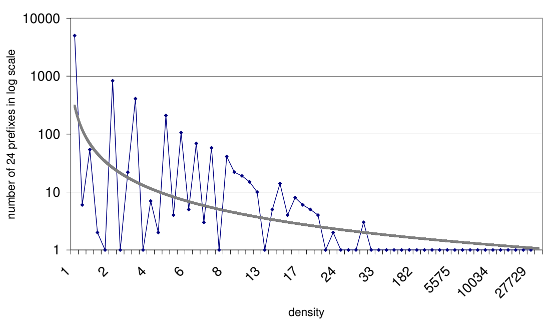
B. Blacklisting

We now present and analyze the results obtained from our blacklisting methodology. We clustered all public IP addresses from our database into /24 prefixes and calculated the density of each prefix. Figures 3(a) and 3(b) show the density distribution for the title *"Pain" by "Jimmy Eat World"* (040). Figures 4(a) and 4(b) show the same plots but for the title *"Let's Get It Started" by "Black Eyed Peas"* (005). For both titles, most of the prefixes have a density of 1. For "Pain", there are 5,012 prefixes with density 1 and for "Let's Get It Started", there are 28,506 such prefixes. It is clear from these figures that for some titles there are prefixes with extremely high densities (there are prefixes with over density of over 10,000 in title 040), while for other titles, all prefixes have low densities(all prefixes have density less than 15 in title 005). It is also clear from these figures that title 005 has not been targeted by the pollution attack whereas the title 040 has.

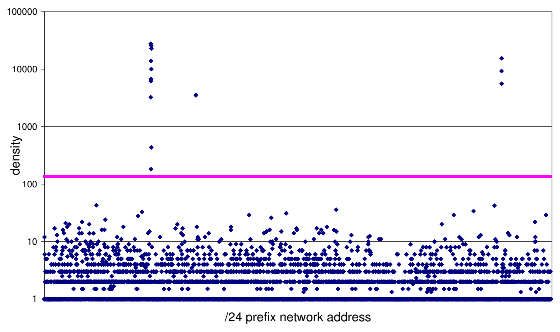
We now turn our attention to determining the blacklist set using the blacklisting methodology developed in Section III. As defined in Equation 1, we use a blacklisting threshold of $kd^{(n)}_{median}$. In that formula k is an experimental constant; through many trials we found that $k = 8$ consistently achieves good separation of ordinary users and polluters. Figures 3(b) and 4(b) include the thresholds for the two titles. In the case of the polluted title "Pain," with median of 17 and resulting threshold of $d^{(040)}_{thresh} = 136$, it successfully manages to separate the majority users from the outstanding few with high density, while in the case of the clean song "Let's Get It Started," with median of 4 and threshold of 32, the threshold is above all prefix densities and thus does not blacklist any prefix.

For the prefixes with densities larger than the threshold for each title, there are 114 /24 prefixes, containing 1,218 IP address (with one of the titles), 70,224,279 title copies, and 10,518,683 versions of 154 of the 170 titles that we had in our crawling database. Note that a very small fraction of the /24 prefixes in FastTrack are responsible for pollution.

The next step of our methodology is the merging of the prefixes that are topologically close. Merging the consecutive /24 prefixes and those that have the same last hop router resulted in decreasing the number of clusters to 101 prefixes, with the masks ranging from /24 to /16. We then performed the BGP prefix verification. The BGP prefixes are from [10] obtained on 12/06/04. We had information about 17,037,611 prefixes. Some of the prefixes that resulted from merging were part of different BGP prefixes. Those merges had to be abandoned. After the BGP verification we had a final list of identified polluter IP ranges, details for which we present in

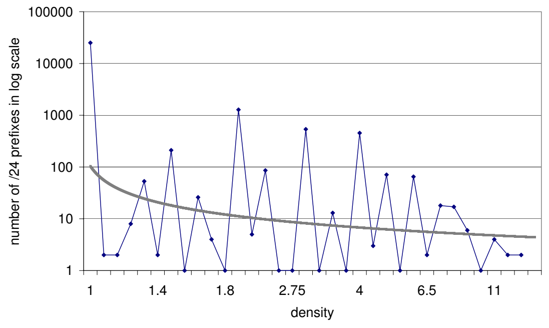


(a) Number of /24 subnets per density value

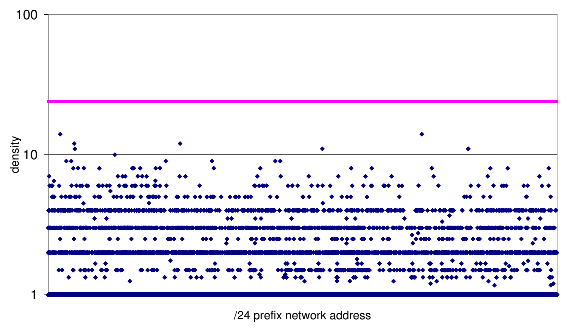


(b) Density distributions of the /24 subnets of the entire IPv4 range and blacklist threshold

Fig. 3. Density plots for title 040 ("Pain")



(a) Number of /24 subnets per density value



(b) Density distributions of the /24 subnets of the entire IPv4 range and blacklist threshold

Fig. 4. Density plots for title 005 ("Let's Get It Started")

TABLE III
BLACKLISTING RESULTS

Nodes	Number of IPs	Number of prefixes	Number of BGP prefixes	Number of BGP ASs
Blacklisted	1218	112	79	59
Non-Blacklisted	1,303,954	325,075	15,747	4,296

TABLE IV
NODE STATISTICS

Nodes	Avg. copies per title	Variance of # of copies	Avg. users per IP
Blacklisted	11.67	20.0	269
Non-blacklisted	1.56	1.88	1.01

Table III. We see from the table that the resulting blacklist set $\mathcal{P}$ has 112 prefixes. These prefixes contain 1,218 IP addresses that contain at least one copy of one of the investigated titles. The table also shows that the methodology does not blacklist the remaining 325,075 prefixes, which contain over 1.3 million IP addresses with the investigated content. To understand better the distribution of those prefixes we also determined the number of BGP prefixes and ASs that the were supersets of the the blacklisted ranges. We present those numbers also in Table III. The 112 prefixes that we found are parts of 79 BGP prefixes, or 59 BGP ASs - a very limited set of prefixes compared to the total number of prefixes and ASs found in the BGP tables.

Table IV provides important insights into the characteristics of the nodes blacklisted by the methodology. A non-blacklisted node, when it has at least one copy of a particular title, has on average 1.56 copies of that title. On the other hand, a blacklisted node, when it has at least one copy of a particular title, has on average more than 11 copies of the title (each of a different version)! The variance of the number of copies per title is also reported in Table IV. Finally, the number of users (client instances) per node is also reported. It is interesting to note that a blacklisted node has on average a remarkable 269 user instances per IP address. In contrast, non-blacklisted nodes have essentially just one instance per IP. The exact value of 1.01 can be explained by the use of SOCKS proxies that allow different users to connect to the P2P network with the same public IP address but different username and port number.

C. Evaluating the Accuracy of the Blacklist

The last-hop RTT experiment was described in Section IV. Fig 5 shows the results of the experiment. We compare the last-hop RTTs of 3,120 randomly chosen non-blacklisted nodes with all 1,218 blacklisted nodes. Since not all of these nodes were up and not all routers replied to the traceroutes, we were able to successfully measure 401 non-blacklisted nodes and 523 blacklisted nodes. Fig 5 shows that the vast majority of the non-blacklisted nodes have a last hop RTT in the 5-15 ms range with average value of 15.27 ms and median of 5.32ms. Over 45% of non-blacklisted nodes have a last-hop RTT below 5ms, while for less then 10% it is over 45 ms. This diversity is quite reasonable and in agreement with previous research [22]. The different values match the different Internet access links that users typically have ($<$ 1ms for LAN, $>$5m for cable, $>$15ms for ADSL and $>$150ms for dial up modem). In contrast, the average estimated last-hop RTT for the blacklisted nodes is 0.67ms, and the median is 0.1ms (typical for LAN connections). Thus, the last-hop RTTs provide evidence that the nodes in our blacklist set are polluters whereas the nodes outside the blacklist set are ordinary users. The average values for both blacklisted and non-blacklisted nodes are listed in Table V.

We now turn to our TCP throughput experiment as described in Section IV. We used a distributed approach to avoid any limits imposed on our campus connection and obtain an average throughput from different geographic locations. We again compare blacklisted nodes with non-blacklisted nodes. Figure 6 shows the CDFs for these two classes of nodes. In these CDFs, the nodes are re-ordered from lowest throughput to highest throughput. We see that more than 95% of the measured non-blacklisted nodes had a throughput less than 20 KBytes/sec. At the same time, more than 95% of the blacklisted nodes have a throughput of more than 20 KBytes/sec. Thus, the TCP throughput provides further evidence that the nodes in our blacklist set are polluters whereas the nodes outside the blacklist set are ordinary users. This observation made us chose the value of 20KBytes/s as a threshold in our false positive and false negative evaluation. The average values for both types of nodes are presented in Table V.

TABLE V
AVERAGE THROUGHPUT AND LAST-HOP RTT

Nodes	TCP Throughput	Last Hop RTT
Blacklisted	2,478 kbps	0.67ms
Non-Blacklisted	75.8 kbps	15.27ms

In order to evaluate the false negatives and the false positives, we use the methodology described in Section III. We set the threshold for the estimated last hop RTT to 1ms and the threshold for throughput to 20KBps.

We tested 1,000 users (with unique IP addresses) from outside of $\mathcal{P}$ for false negatives. 28 had more than 5 versions of a title and passed the first test. We randomly downloaded 5 versions from each of those users and determined that for 4 of the users all versions were corrupted. We finally applied the TCP throughput and last hop RTT tests. Only 2 users failed all 4 tests. Thus, the false negative ratio of the blacklisting methodology can be estimated to 0.2%.

We also tested 1,000 users from inside of $\mathcal{P}$ for false positives. After randomly downloading 5 versions for content testing, we found that 46 users provide at least one non-polluted version of a title. The next test determined that 38 and 25 users failed the throughput and last hop RTT tests

TABLE VI
BLACKLIST EVALUATION TESTS

	Versions	Polluted	Throughput	RTT	Total
False neg	2.8%	0.40%	7.5%	12%	0.20%
False pos	N/A	4.60%	3.8%	2.5%	7.10%

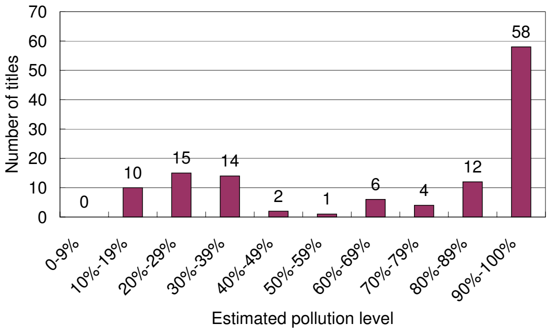


Fig. 8. Estimated Pollution Level of all 122 analyzed titles

respectively. Overall, 71 users failed at least one of the 3 tests causing a false positive ratio of 7.1%. We suspect those to be regular KaZaA clients that the polluters use to study the network and their targets. Details on the false positive and negative results are shown in Table VI

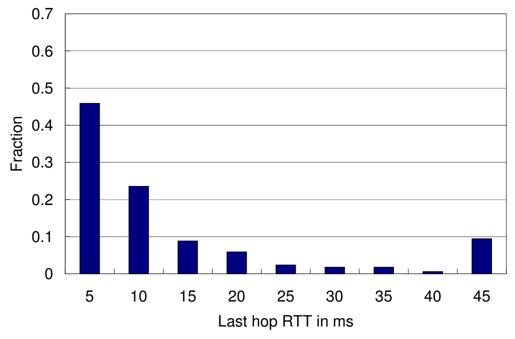
We also evaluate the effect of the blacklisting methodology by comparing the average number of copies per user in FastTrack without blacklisting and with blacklisting. Figure 7(a) shows a plot of the average number of copies and the variance for the 122 titles that we analyzed when blacklisting is employed. Fig 7(b) shows the same graph without the blacklisting. Note that the scale of the graph changes by a factor of 20 and becomes much more uniform. After blacklisting, the average number of copies of a title dropped down to less than 3 for all titles with a maximum variance of about 3.

D. Estimating Pollution Levels

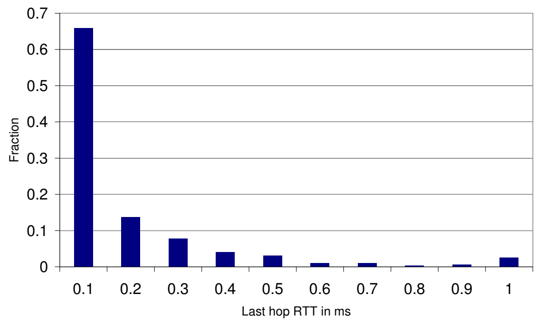
We used our methodology described in Section III to determine the pollution levels of the 122 investigated titles.

We present the pollution levels, obtained by Formula 2 in Section III, for all 122 titles. Since this would take up too much space, in Figure 8 we present a bar graph showing the number of titles with their pollution level in different intervals. The picture shows that 58 of the titles have estimated pollution level of 90% or more, while 51 titles have pollution level of less than 50%. There are no titles with pollution level of 0% because every title has some number of polluted versions out there.

To give further insight into pollution levels, we plotted the CDFs of the fraction of copies versus the fraction of users for the titles under investigation. Due to space constraints, in Fig 9(a) we only show the plots for 8 representative titles. For some titles the large percentage of all copies is concentrated within a small number of users; those titles have a very skewed CDF. Other titles have a more uniform distribution for the number of versions per user. The CDF of a title gives a visual

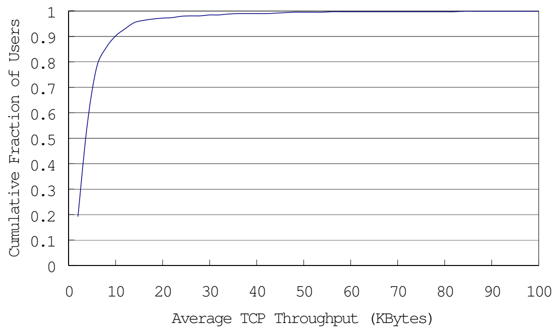


(a) Last-hop RTTs for non-blacklisted nodes

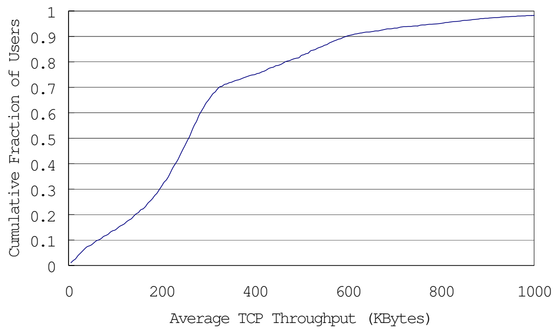


(b) Last-hop RTTs for blacklisted nodes

Fig. 5. Last-Hop RTTs

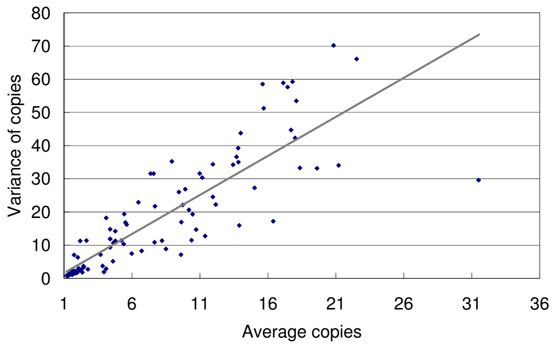


(a) Average throughput of non-blacklisted nodes

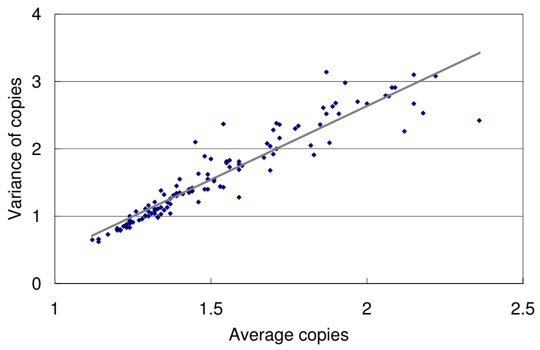


(b) Average throughput of blacklisted nodes

Fig. 6. Average TCP Throughput



(a) Before Blacklisting



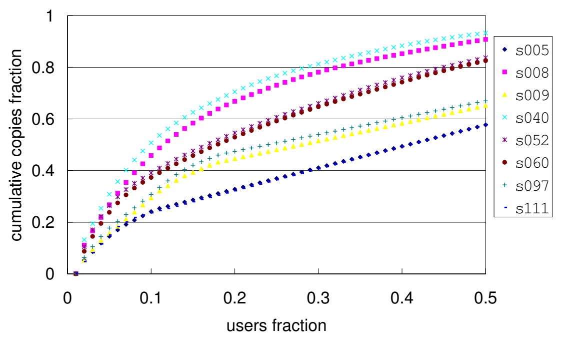
(b) After Blacklisting

Fig. 7. Average number of copies per user and variance for all 122 titles

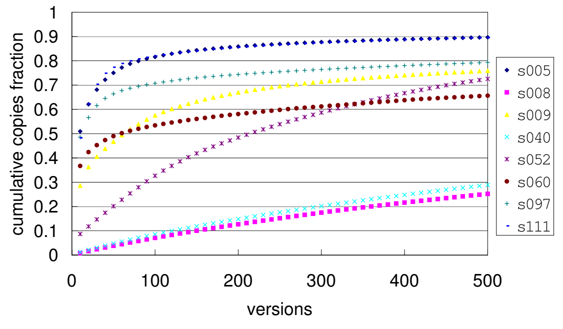
representation of its pollution level. A very skewed CDF shows that just a few users have copies of most of the versions of a title. A regular user does not normally have hundreds or thousands of versions of the same song; so those users must be polluters and the title must be polluted.

Another interesting result that we present here is the distribution of the number of copies of a title and the number of versions. The CDFs for the 8 representative titles are shown in Figure 9(b). The figure shows that for some titles the top 100 versions account for 80% or more of the copies that are available on the network. This result is indeed expected since clean songs usually have few popular versions. Other titles, however, are highly scattered, having as many as the top 500 of their versions accounting for a less than 30% of the total number of copies for that title. This also matches our expectations and explains why it is very difficult to find a clean version of a highly-polluted title. Thus, the highest polluted title on the plot is 008, while the cleanest one is 005.

Fig 10 reflects the pollution level contributed only by the P2P users with public vs. users with private IP addresses. We concentrate on the public users because we expect to find the dedicated polluters in the non-NATed ranges of IP addresses.



(a) Copies per user



(b) Copies per version

Fig. 9. CDFs for 8 Titles

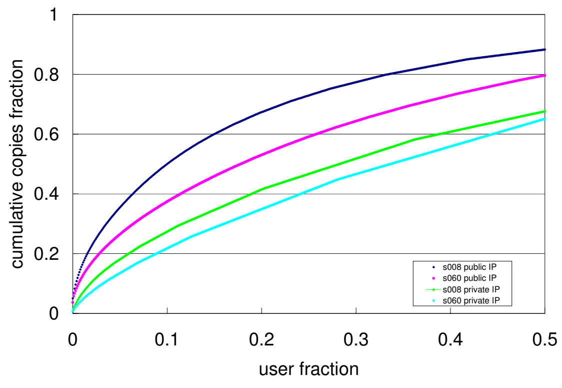


Fig. 10. CDF of copies per user for only public IP addresses and only private IP addresses for 2 polluted titles

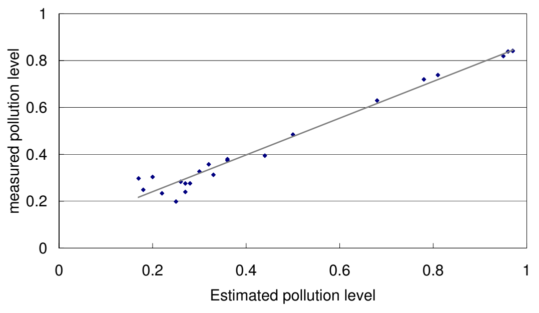


Fig. 11. Measured vs. estimated pollution levels for 24 titles

If, however, we look at private IP addresses and their CDF, we can see a distribution that is guaranteed to have no dedicated polluters. That, however, does not mean that there are no polluted copies of content in private IP users. Because the number of copies provided by public IPs is much bigger than the number provided by private IPs the influence of NATed users to the pollution level of a title is insignificant. This can be easily shown by observing that the skewness of the CDF for public users on Fig. 10 is essentially the same as that in Fig. 9(a) which includes both public and private users.

Our results indicate that polluted songs can be separated in 3 different groups. We define the 3 stages of Pollution Evolution and give their specific characteristics in Section II-C. Table VII displays detailed information about the titles from all three evolution stages. It also provides information for two representative clean titles. In this figure PU and OU signifies "polluting user" and "ordinary user," respectively. Each consecutive stage is characterized by lower polluter contribution in the number of users per IP address, average number users, and average number of copies. The variance in the number of copies also decreases significantly.

E. Pollution Level Evaluation

We now evaluate our pollution level estimates, using the procedure described in Section 3. Recall that this procedure compares our estimated pollution level $E^{(n)}$ with the measured pollution level $L^{(n)}$. We selected 24 songs for which the top 200 versions represented 80% or more of the total number of copies. We then downloaded the top 200 versions and used our automatic testing procedure to determine if they are polluted or not. Figure 11 shows the correlation of the measured and estimated pollution level for those titles. The plot is consistently linear and indicates that our procedure for estimating the pollution levels of titles is quite accurate. The 10% difference in the correlation can be explained by the fact that we don't actually download all the versions but just the top 80% or so (the top 200 versions sometimes correspond to more than 80%). We then computed the value of the error for the estimated pollution level as discussed in our Methodology (Equation 4) and its value was 6.8%.

VI. Related Work

Although a relatively new Internet application, there are many measurement studies on P2P file-sharing systems. Bandwidth, availability, and TCP connection duration for popular filesharing systems such as Gnutella and Napster are examined in [11] [12][14] [3]. P2P user behavior and content characteristics are studied in [13] [15].

TABLE VII
CONTENT METRICS

Pollution status	Title	Number of users		Avg. copies per IP		Variance of copies		Polluter copies
		PU	OU	PU	OU	PU	OU	
Introductory	040	121975	13127	8276	1.99	53.7	2.99	0.987
	008	103583	8959	2808	2.16	45.45	2.96	0.988
Growth	060	84885	87994	422	1.43	9.95	1.35	0.59
	052	41442	84784	420	1.38	1.33	1.16	0.45
Decline	097	194	28389	17.6	1.32	0	1.07	0.005
	009	448	40957	9.1	1.34	0.23	1.13	0.009
Post-Pollution	111	2	14517	1	1.24	0	0.81	0.0003
	005	7	56344	1	1.21	0	0.79	0.0001

Several independent research groups have developed crawlers for P2P file sharing systems. A crawler for the original single-tier Gnutella system is described in [3]. A crawler for the current two-tier Gnutella system (with "ultrapeers") is described in [4]. A crawler for eDonkey is described in [15]. A crawler for the FastTrack P2P file sharing is described in [2].

There is related work on attacks and shortcomings of P2P systems. The freerider problem, potential attacks to and from P2P systems, and the DRM are considered in [16], [17] and [18]. DoS attacks in P2P systems are investigated in [16] [19]. Viruses are addressed in [16] and [20].

There are also relevant related studies of CDNs and peer selection. RTT bandwidth, and TCP throughput are examined in [21] for server selection purpose and RTT, throughput probing and bandwidth measurement are considered in [22] for heterogenous P2P environments. In particular, our throughput and last-hop RTT techniques were derived from [22].

In [2] it was established that pollution is widespread for popular titles. The methodology used in [2] to determine pollution levels is inefficient in that it requires downloading and binary content analysis of hundreds of versions for every investigated title. Although the current paper makes use of the FastTrack crawler in [2], the contribution is very different. It is the first paper to develop a blacklisting methodology for file-sharing systems. Furthermore, all the methodologies in this paper do not require any downloading and are solely based on metadata gathered from the crawler. A user-supported antip2p banlist is available in [23].

VII. CONCLUSION

In this paper we considered two related problems: creating blacklists for polluting IP address ranges, and estimating the pollution level of targeted titles. Adequate solutions to both of these problems require accuracy and efficiency. Extensive tests with data collected from the FastTrack P2P file-sharing system have shown that our methodologies meet both of these goals.

Our methodologies do not involve the downloading of any files. Instead, they identify polluting IP address ranges and targeted titles by collecting and analyzing metadata from the file sharing system. The metadata is harvested by crawling the nodes in the file sharing system. From this harvested metadata, our methodology constructs the blacklisted IP address ranges and the estimated pollution levels for the targeted titles. The methodology is efficient in that it collects metadata (text) rather than binary content and that a large number of titles, and virtually all the file-sharing nodes can be investigated in one crawl.

To address accuracy, we developed several criteria to evaluate the methodologies. We then applied these criteria to a comprehensive test case for the FastTrack file-sharing system. For blacklisting, we found the probability of false negative and false positive to both be low, namely, 0.2% and 7.1%, respectively. After applying the blacklist, the average number of versions per user for polluted titles decreases dramatically. These results testify to the overall accuracy of our blacklisting methodology. For estimating pollution-levels, we compared our estimated pollution-levels with measured estimates, which involved the downloading and binary analysis of titles. For our comprehensive test case, we found the percentage error to quite low, less than 7%.

In a real deployment, it is important that the blacklist set and the pollution-level estimates adapt as polluters change hosts and target new content. Our methodology is naturally suited for such a dynamic environment. The crawler can operate continuously, collecting metadata for fresh titles as they become released. The fresh titles can be obtained directly from on-line billboard charts. Similarly, our methodology can continuously be applied to the data in the crawling database, thereby dynamically adjusting the blacklist set and the pollution-level estimates.

REFERENCES

[1] CacheLogic Research: The True Picture of P2P File Sharing, http://www.cachelogic.com/research/

[2] J. Liang, R. Kumar, Y. Xi, K. Ross. Pollution in P2P File Sharing Systems, *Infocom 2005*

[3] 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.

[4] D. Stutzbach, R. Rejaie. Characterizating Today's Gnutella Topology, *submitted*.

[5] R. Schemers, fping utility, http://www.fping.com/

[6] Apple iTunes Top 100, http://www.apple.com/itunes/

[7] Top 100 songs from 1970 to 1979, http://top40.about.com/cs/70shits/

[8] Blockbuster's Top 100 Online Rentals, http://www.blockbuster.com/

[9] Z. M. Mao, J. Rexford, J. Wang, and R. Katz, Towards an Accurate AS-Level Traceroute Tool, *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003

[10] CIDR Report, http://www.cidr-report.org
[11] S. Sen, J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks, *ACM/IEEE Transactions on Networking, Vol. 12, No. 2, April 2004*
[12] S. Saroiu, P. K. Gummadi, S. D. Gribble, A Measurement Study of Peer-to-Peer File Sharing Systems, *Multimedia Computing and Networking (MMCN'02)*, San Jose, January 2002
[13] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, 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*
[14] V. Aggarwal, S. Bender, A. Feldmann, A. Wichmann. Methodology for Estimating Network Distances of Gnutella Neighbors. *Proceedings of the Workshop on Algorithms and Protocols for Efficient Peer-to-Peer Applications at Informatik, 2004*
[15] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, L. Massouli. Clustering in Peer-to-Peer File Sharing Workloads. *IPTPS'04*
[16] D. Katabi, B. Krishanmurthy. Unwanted Traffic: Attacks, Detection, and Potential Solutions. *ACM SIGCOMM'04 Tutorial*
[17] P. Biddle, P. England, M. Peinado, and B. Willman. The Darknet and the Future of Content Distribution. *ACM DRM 2002*
[18] M. Feldman, C. Papadimitriou, J. Chuang, I Stoica. Free-Riding and Whitewashing in Peer-to-Peer Systems. *ACM SIGCOMM'04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), August 2004*
[19] S. Dropsho. Denial of Service Resilience in Peer to Peer File Sharing Systems.*EPFL Tech Report*
[20] http://www.bullguard.com
[21] S. G. Dykes, K. A. Robbins, C. L. Jeffery. An Empirical Evaluation of Client-side Server Selection Algorithms. *Infocom 00*
[22] T.S.E Ng, Y. Chu, S.G. Rao, K. Sripanidkulchai, H. Zhang. Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems. *IEEE Infocom'03*
[23] Bluetack Internet Security Solutions, http://www.bluetack.co.uk/
[24] The FFmpeg project, http://ffmpeg.sourceforge.net/index.php
[25] B. Krishnamurthy, J. Wang. On network-aware clustering of Web clients. *ACM SIGCOMM 2000*