Content Availability, Pollution and Poisoning in File Sharing Peer-to-Peer Networks*

Nicolas Christin
S.I.M.S., UC Berkeley
christin@sims.berkeley.edu

Andreas S. Weigend
Weigend Associates LLC
andreas@weigend.com

John Chuang
S.I.M.S., UC Berkeley
chuang@sims.berkeley.edu

ABSTRACT

Copyright holders have been investigating technological solutions to prevent distribution of copyrighted materials in peer-to-peer file sharing networks. A particularly popular technique consists in "poisoning" a specific item (movie, song, or software title) by injecting a massive number of decoys into the peer-to-peer network, to reduce the availability of the targeted item. In addition to poisoning, pollution, that is, the accidental injection of unusable copies of files in the network, also decreases content availability. In this paper, we attempt to provide a first step toward understanding the differences between pollution and poisoning, and their respective impact on content availability in peer-to-peer file sharing networks. To that effect, we conduct a measurement study of content availability in the four most popular peer-to-peer file sharing networks, in the absence of poisoning, and then simulate different poisoning strategies on the measured data to evaluate their potential impact. We exhibit a strong correlation between content availability and topological properties of the underlying peer-to-peer network, and show that the injection of a small number of decoys can seriously impact the users' perception of content availability.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks

General Terms

Measurement, Performance, Reliability

Keywords

Peer-to-peer networks, File sharing, Content protection

1. INTRODUCTION

Since its inception in 1999 with the Napster service, peer-to-peer file sharing has grown to the point of becoming one of the

predominant sources of Internet traffic [21, 23]. As a result, and even though the actual impact of peer-to-peer file sharing on product sales remains hard to assess (e.g., [20] and [25] reach opposite conclusions), copyright holders are now more than ever worried by the potential loss of revenues due to file sharing, and have been exploring several options for thwarting file sharing in peer-to-peer networks. In particular, while legal action, for instance the case against Napster [1], has received the most significant exposure in the popular press, considerable effort has also been devoted to investigate technological solutions for reducing content availability in peer-to-peer file sharing systems.

A common technique to decrease the availability of a specific item (e.g., movie, song, software distribution) in a peer-to-peer network consists in injecting a massive number of decoys into the network [13]. The decoys are files whose name and metadata information (e.g., artist name, genre, length) match those of the item, but whose actual content is unreadable, corrupted, or altogether different from what the user expects. For instance, many peer-to-peer users who tried to download the song "American Life" by Madonna found themselves in possession of a track that only contained a message from the artist chiding them for using file sharing services. We refer to such a deliberate injection of decoys as *item poisoning*.

In addition to poisoning, the accidental injection of "bad" (i.e., poorly encoded or truncated) copies of files in the network, or *network pollution*, also has the effect of decreasing the proportion of usable content in the network. For instance, a recent study [17] provides empirical evidence that a substantial fraction of the files served in the KaZaA/FastTrack network are unusable, due to either pollution or poisoning.

However, while pollution and poisoning both result in introducing unusable files in the network, their respective characteristics and impact are significantly different. More precisely, pollution can generally be assimilated to (almost) random noise, whereas poisoning aims at changing the availability of a specific item in the network, by deliberately making it harder to find usable copies of the item.

In this paper, we make a first step toward understanding the impact of pollution and poisoning on content availability in peer-to-peer file sharing networks. We notably investigate questions such as "which level of network pollution is really harmful?" or "is a given poisoning strategy effective at limiting the availability of the item it targets?"

Our specific contributions are as follows. We first provide a measurement study of content availability in the four most popular (at the time of this writing) peer-to-peer file sharing networks, in the absence of (blatant) poisoning. We next discuss the differences between network pollution and possible poisoning strategies, some of which have been observed in practice [17]. We then evaluate the

effect of network pollution and poisoning on content availability, by numeric simulation on the gathered measurement data. We exhibit a potentially strong correlation between content availability and topological properties of the underlying peer-to-peer network, and show that the injection of a small number of decoys can seriously impact the users' perception of content availability.

As a caveat, we point out that this paper solely focuses on the properties of the file sharing *networks*. More precisely, while we do look at metrics that influence user behavior, such as the time to complete a download, we defer the study of actual user behavior (e.g., through laboratory experiments with human subjects) to future work.

The remainder of this paper is organized as follows. In Section 2, we briefly review some of the related measurement studies that have been proposed in the literature. In Section 3, we summarize how the various peer-to-peer networks we investigate respond to user queries. In Section 4, we report our measurements of content availability in the four most popular peer-to-peer networks. In Section 5, we use the measurement data obtained to characterize by simulation the response of the networks under consideration to pollution and to different types of poisoning attacks. Finally, in Section 6, we draw brief conclusions and identify some avenues for future research.

2. RELATED WORK

The rapid rise of peer-to-peer systems has prompted number of quantitative works. Some studies, e.g. [14, 21, 23], take a bird's eye view of commercial or university networks, and assess the impact of peer-to-peer traffic on the underlying physical network. In particular, Saroiu et al. [21] provide convincing evidence of the very high level of peer-to-peer traffic in university campuses, and Karagiannis et al. confirm in [14] that the amount of peer-to-peer traffic is not declining, despite the growing legal threats on peer-to-peer users.

Other measurement works investigate topological properties of peer-to-peer systems. For instance, Liang et al. discuss properties of the KaZaA/FastTrack network in [16], Loo et al. describe the evolution of the Gnutella topology in [18], and Tutschku characterizes eDonkey traffic in [24]; Saroiu et al. [22] exhibit a high heterogeneity in the hosts connected to the Gnutella and Napster networks, while Bhagwan et al. [5] look at peer availability, and notably at the turnover rate of Overnet hosts.

A few studies measure content location and popularity in peer-to-peer networks. Chu et al. [6] exhibit power-laws in content replication in the Napster and Gnutella networks. Gummadi et al. [12] show that, on the other hand, download requests significantly deviate from a power-law distribution, because most users download files only once. Le Fessant et al. [15] show that the eDonkey network presents geographical clustering properties, which could be taken advantage of with the appropriate content replication algorithms.

All of these works provide us with a very good understanding of the properties of peer-to-peer file sharing systems at the network level, by mostly relying on passive measurements; that is, they monitor the network without introducing noticeable perturbations. Because we are more concerned in how end users perceive the network, we use active measurements, which consist in presenting the network with an input, and measuring the response of the network to that input.

In that respect, Liang et al.'s study [17] is more closely related to our study. Liang et al. send a set of queries into the FastTrack network, and measure returns to their queries. They show they obtain a substantial proportion of incomplete or corrupted files, and provide a methodology to automatically assess whether a file is a decoy. Our study takes a different, and complementary, approach, by making the distinction between pollution and poisoning, and evaluating the potential impact of different poisoning strategies. Additionally, we not only investigate the FastTrack network, but also examine the properties of the eDonkey, Overnet, and Gnutella networks.

Last, in a study conducted simultaneously and independently of the work described in this paper, Dumitriu et al. investigate possible attacks on peer-to-peer file sharing systems by mathematical modeling and simulation [10]. Our study, on the other hand, relies on measurements of field data, and focuses on poisoning attacks that aim at discouraging users from downloading a specific file, rather than on attacks that attempt to bring an entire peer-to-peer system down.

3. BACKGROUND

As evidenced by the demise of the Napster network, which quasi-immediately followed the shutdown of the search infrastructure, the success of a peer-to-peer network is generally driven by content availability. Content availability describes how easily content can be found and downloaded, and is itself directly conditioned by the network response to user search queries.[1] How queries are processed is itself highly dependent on the topology of the peer-to-peer network, which we discuss in this section.

Older peer-to-peer file sharing networks such as Napster relied on a global index of the network contents, hosted on a centralized server. Because one can take down the entire network by attacking the centralized server, as was the case with the legal attack on Napster [1], most of the peer-to-peer networks have since then abandoned a completely centralized search index in favor of distributed search primitives.

In particular, the three most popular peer-to-peer networks, that is, the eDonkey, FastTrack, and Gnutella networks, which have approximately between 1,000,000 and 3,000,000 users each,[2] all rely on two-tiered hierarchical topologies, where nodes are split between leaf nodes and hubs (called "ultrapeers" in Gnutella, "supernodes" in FastTrack, and "servers" in eDonkey). Leaf nodes maintain a connection to a handful of hubs, while hubs maintain connections with hundreds or more of leaves, and with many other hubs. Each hub serves as a centralized index for the leaf nodes that it is connected to. Whenever a leaf node issues a query, the query is sent to the hub(s) the leaf node is connected to. If the item requested is not present in the index maintained by the hub(s), the query is forwarded to other hubs.

The main differences between the eDonkey, FastTrack and Gnutella networks reside in (1) the proportion of hubs among all nodes, (2) the rate at which connections between leaves and hubs change, and (3) the criteria that preside over the promotion of a leaf node to hub status. Different networks also use different formats for query messages, but differences in message formats have generally limited incidence on the number and content of responses to a query, thus we will not discuss them any further here.

We summarize the hierarchical properties of the different networks under study in Table 1. The number of hubs is evaluated using publicly available statistics for eDonkey,[3] and using mea-

---

[1] A notable exception is BitTorrent [9], which does not provide any search facility. As such, BitTorrent is arguably more of an extremely efficient distributed algorithm for downloading a given file, than a peer-to-peer network containing a collection of files.

[2] Data reported as of February 18, 2005 on `http://www.slyck.com`.

[3] `http://ocbmaurice.dyndns.org/pl/ed2k_stats.pl`

	eDonkey	FastTrack	Gnutella
Nr. of hubs	40–90	25,000–40,000	10,000-100,000
Nr. of nodes	$\approx 2.8 \times 10^6$	$\approx 2.5 \times 10^6$	$\approx 10^6$
Frac. of hubs	$\approx 2 \times 10^{-5}$	$\approx 1.5 \times 10^{-2}$	$\approx 5 \times 10^{-2}$
Average leaf-hub connection lifetime	≈ 24 hr	≈ 30 min	≈ 90 min
Leaf promotion	Voluntary	Election	Election

Table 1: Topological characteristics. The table illustrates the differences in topology between the different networks.

Queries Network	Songs 1–2, Movies 1–2, Software 1	Songs 4–5, Movies 4–5, Software 2	Songs 5–6, Movies 5–6, Software 3
Gnutella	6	6	6
eDonkey	6	6	6
eD/Overnet	6	6	6
FastTrack	12	12	12

Table 2: Experimental setup. The table describes the number of hosts on each network that were used to issue each query.

surements presented in [16] and [18] for FastTrack and Gnutella, respectively. Dividing the number of hubs by the total size of the network, we can infer the fraction of hubs in the network. We further use measurements from [16, 18] as well as our own measurements (for eDonkey) to determine the average lifetime of a leaf-hub connection. Note that we only present estimates of averages over all nodes here. While averages are useful to infer general trends, results for specific nodes can significantly deviate from the average, and we refer to [16, 18] for more comprehensive data. These average numbers allow us to make the key observation that eDonkey is much more centralized than FastTrack or Gnutella, relying on a few hubs (servers), and connections between leaf nodes and servers that are much more persistent.

The insight behind the difference in topologies lies in how nodes are promoted from leaf to hub. Promotion is purely voluntary in eDonkey: users interested in hosting a server have to install and run specific server software. Hence, servers are expected to have very long uptimes, a (quasi-)permanent connection to the network, and the ability to handle large number of requests. Conversely, in both FastTrack and Gnutella, leaf nodes are promoted to hubs by the software client, and generally unbeknownst to the user. Even though criteria for promotion to hub status include node uptime, network capacity and processing power, FastTrack and Gnutella hubs exhibit rates of connection and disconnection to the network only slightly lower than those of leaves, and certainly much higher than those of eDonkey servers.

Last, the fourth most popular file sharing network, Overnet, accounts for about 1,000,000 users. Overnet does not distinguish between leaves and hubs, and instead relies on the Kademlia distributed hash table [19] to locate content. However, all Overnet clients simultaneously connect to the eDonkey network,[4] so that we expect to observe substantial content overlap between the eDonkey and Overnet networks.

4. CONTENT AVAILABILITY

Ideally, each node participating in a peer-to-peer network should have the same, global, view of the entire content available on the

---

[4]Clients solely connecting to the Overnet network were only available as "beta" versions, and were discontinued in August 2004.

network, irrespective of time or location. In practice, responses to a query may considerably differ depending on the hub responding to the query. In networks where connections between leaves and hubs are highly dynamic, and with high turnover rate among the peers [5, 16, 18], a user's view of the available content may drastically depend on time and location.

In this section, we outline the differences in (perceived) content availability across different networks, and correlate them with differences in the network topologies. The goal is to gain a better understanding of the factors that influence the sensitivity of a network to poisoning and pollution. To that effect, we conduct a measurement study of content availability in the eDonkey, eDonkey/Overnet, FastTrack, and Gnutella networks *in the absence* of observable poisoning, so that we can later (in Section 5) separately characterize the effects of different poisoning strategies on each network. We next motivate and discuss our measurement infrastructure, describe our experimental methodology, and report our observations.

4.1 Measurement infrastructure

Logical overlay network topologies such as peer-to-peer networks generally bear little resemblance to the underlying geographical locations of their participants. However, we conjecture that peer-to-peer nodes located in geographically distant areas are unlikely to be topologically close in the peer-to-peer network. Thus, we try to obtain a global view of the networks under consideration, by running peer-to-peer clients on a number of geographically dispersed nodes in the PlanetLab infrastructure [7]. We run peer-to-peer clients on over 50 nodes located in 18 different countries in North and South America, Europe, Asia, and Oceania. PlanetLab nodes connect to the Internet through different ISPs and different types of physical links, including broadband access (DSL).

We use MLDonkey [4] to connect to the eDonkey, eDonkey/Overnet,[5] and Gnutella networks, and giFT-FastTrack [2] to access the FastTrack network. The main advantage of MLDonkey and giFT-FastTrack is that both implement daemons that can be accessed through `telnet`-based interfaces. Hence, experiments are easily scriptable, and therefore easily repeatable. We communicate with the daemons using simple Perl clients to search and download files in all four networks. As an aside, nodes under our control only implement leaf functionality, and cannot be used as a hub. In other words, none of our nodes is a FastTrack supernode, a Gnutella ultrapeer, or an eDonkey server. Because we are more interested in how users see the network rather than considering aggregate of requests, this limitation does not affect our study.

4.2 Experimental methodology

As we mentioned earlier, active measurements are a good fit for our approach, since we want to contrast the response of the network depending on whether or not the network is subject to poisoning. In addition, the most popular items on the network are likely to be poisoned. Therefore, poisoning could account for a vast majority of the traffic observed using passive measurements, ultimately making the distinction between poisoning effects and usual network behavior difficult.

The main drawback of active measurements is that results can heavily depend on the nature of the input we inject in the network. In other words, we have to find a set of queries that are representative enough to give us an accurate picture of the network. In an effort to cover the three main categories of content available in peer-to-peer file sharing networks, we choose 15 query strings cor-

---

[5]Like the official Overnet client, MLDonkey requires to simultaneously connect to the eDonkey network to access the Overnet network.

	eDonkey			eDonkey/Overnet			FastTrack			Gnutella		
	Songs	Movies	Soft.	Songs	Movies	Soft.	Songs	Movies	Soft.	Songs	Movies	Soft.
Avg. number of responses	648	369	790	759	473	909	32	6	348	68	186	563
(Std. dev.)	(292)	(210)	(237)	(315)	(236)	(200)	(37)	(7)	(291)	(76)	(185)	(528)
Avg. number of unique files	578	282	588	668	348	650	22	4	178	65	179	521
(Std. dev.)	(268)	(163)	(166)	(294)	(179)	(106)	(23)	(4)	(123)	(72)	(178)	(492)

Table 3: Number of query returns. The table provides both the total number of query returns and the number of unique files returned. Numbers correspond to the number of returns obtained after 10 minutes for Gnutella, FastTrack and eDonkey.

responding to 6 movies, 6 popular songs, and 3 popular software titles. (To avoid facilitating potential copyright infringement, we refer to the different queries as Song 1 through 6, Movie 1 through 6, and Software 1 through 3, respectively.) We use "specialized" queries for songs and movies to improve the quality of the search returns; that is, we restrict the possible returns to MP3 files and video files, respectively.

For each of the 15 queries, we manually verify that the item queried is not subject to poisoning (or at least, that a potentially ongoing poisoning attack has negligible effect); that is, we check that a few "good" files can be easily found and downloaded. On the other hand, we cannot guarantee the network is not subject to pollution; in fact, we experience various pollution levels depending on the network and query considered, as we discuss later.

We inject the queries in each network as described in Table 2. A bug in MLDonkey causes the results of concurrent queries on a same host to be sporadically mixed, so we run only one ML-Donkey client per host, and group queries into three groups of five queries (2 songs, 2 movies, and 1 software distribution) each. For each group of the three groups of queries, we send the queries from 6 hosts connected to the Gnutella network, 6 hosts connected to the eDonkey network, and 6 hosts connected to the eDonkey/Overnet network. In addition, we also issue the queries on 12 hosts connected to the FastTrack network. On each host, we repeatedly issue the queries every half-hour for 36 hours.

Last, when a peer-to-peer client is first installed and run on a host, it uses a bootstrapping mechanism that typically results in connecting to a fixed, well-known set of hubs. We attenuate the impact of the initial bootstrapping mechanisms on our experimental results by running the clients for several days before starting to collect data. More precisely, with the exception of one experiment (as discussed later), all clients were started between November 26 and 27, 2004, and all data presented in this paper was collected over December 1–5, 2004. The length of the collection period allows us to circumvent transient and short-term effects, such as time-of-the-day dependency; a comparison with previous experiments conducted over October 7–14, 2004, and which we do not report here, indicates that seasonal effects do not play a substantial role in the set of measurements we are gathering.

4.3 Experimental results

All network properties have, to some extent, an impact on how people exchange content on peer-to-peer file sharing networks. Because we do not directly study user behavior, we have to find the set of network metrics that are likely to have the most impact on users' decisions to use or instead abandon a given network. While we do not claim the metrics we select describe exhaustively all factors that condition user behavior, we focus on a set of five metrics that intuitively play a key role in how peer-to-peer users perceive a network: number of responses to a query, response time to a query, content stability, content replication, and download completion time.

Number of query returns Table 3 provides the average number of responses to our queries we obtained for each network 10 minutes after having issued the query, averaged over all songs, movies and software titles. Because a given file may be hosted on several peers simultaneously, we distinguish between the total number of responses and the number of unique files returned. We make several observations. First, we have significantly more returns in eDonkey and eDonkey/Overnet than in the other networks. This does not necessarily imply that the eDonkey network has more content available than the other networks. In fact, a more likely cause for the observed difference is that each hub in FastTrack and Gnutella indexes the contents of a much lower fraction of the total number of nodes than in eDonkey. Thus, each node in FastTrack and Gnutella has a relatively limited search horizon, which results in lower numbers of returns, and in the returns being more sensitive to nodes leaving and joining. The high variability in the observed number of query returns in FastTrack and Gnutella seems to confirm our hypothesis. In addition, we notice that specialized searches (movies and songs) in FastTrack result in a low number of returns. This can be due to either high levels of pollution (specialized searches tend to filter out some of the polluted items), or to a bug in how the giFT-FastTrack daemon handles specialized searches. We need further measurements, some of which we discuss later, to clarify the possible causes.

Query response times Because searches are not fully centralized, different query results are returned to the sender at different times. Query results that arrive quickly are more likely to be selected for download by most users, who generally have limited patience. Hence, the distribution of the query response times (that is, the time difference between a query is issued and a specific return reaches the sender) plays an important role with respect to the users' perception of content availability.

We plot the distribution of the query response times for all four networks in Fig. 1. The thin lines in the plots show the average over all queries of each type (songs, movies, and software titles). A better indicator might be the 90th percentile of all queries (thick lines), which provides an upper bound for the query response times experienced by 90% of the queries. We observe that eDonkey and eDonkey/Overnet produce results extremely quickly: after two minutes, for nearly all queries, the sender has received over 85% of all query returns. After 3.5 minutes, the network has returned virtually all responses to every query. We can explain this small response time by the highly centralized topology in eDonkey: the first server to be contacted already has most of the results available. In fact, the couple of jumps one can observe in each of the plots in Figs. 1(a) and (b) correspond to results coming from different eDonkey servers. Conversely, Gnutella seems to produce results almost continuously, and FastTrack exhibits a long-tailed distribution of query response times for software titles. These results indicates that queries are propagated to many different hubs
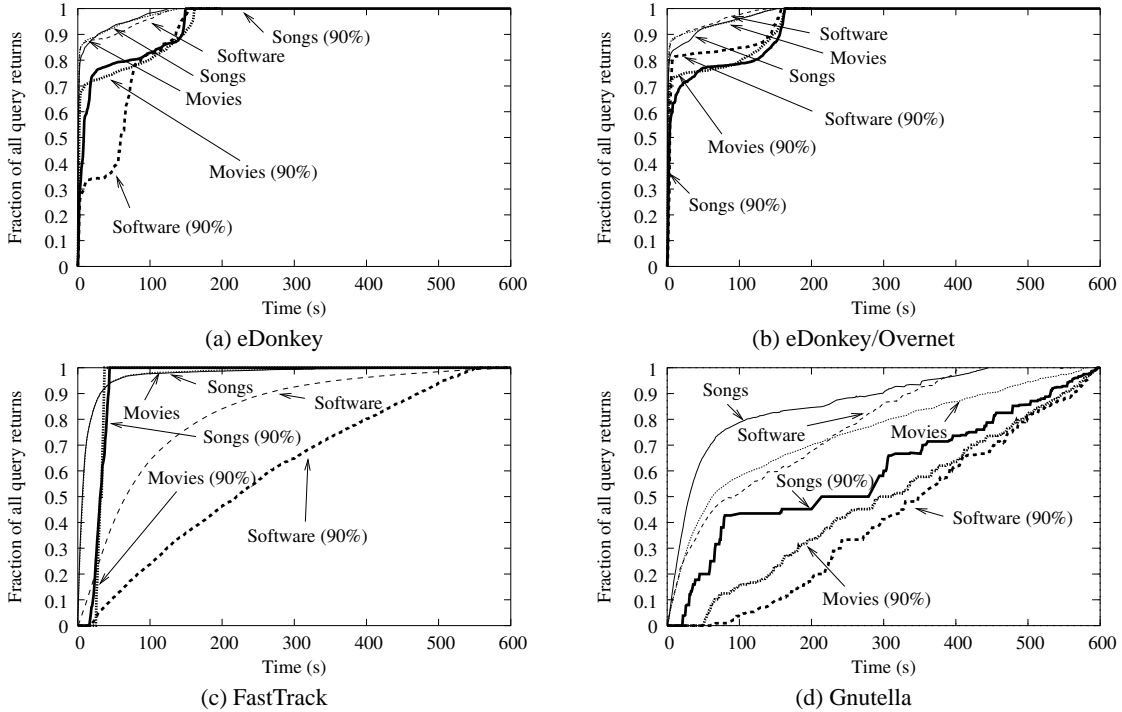
Figure 1: Query response times. The plots describe the average (thin lines) and 90th percentile (thick lines) of the query response times (normalized over the final number of returns), for all three types of queries in the four networks under consideration.

that answer to the sender at different times. We note that FastTrack seems to respond very quickly to specialized searches (movies and songs). We speculate that the specialized searches were not propagated to other hubs, which would explain both the low number of returns we observed in Table 3, as well as the very quick response time.

Content stability We use a time-dependent function we call *temporal stability*, χ, to assess how the users' perception of the available content changes over time. Denoting by $U(t)$ the set of query returns corresponding to unique files returned at time t, we define $\chi(\tau)$, for any $\tau \in \mathbb{R}$, as

$$\chi(\tau) = \frac{\sum_t |U(t) \cap U(t+\tau)|}{\sum_t |U(t) \cup U(t+\tau)|} \ .$$

In other words, $\chi(\tau)$ is the average probability (averaged over all times) that an item returned at a given time T is also returned at time $T + \tau$, for any τ. We always have $\chi(0) = 1$, and values of $\chi(\tau)$ for $\tau < 0$ characterize the probabilities an item returned at a given time had also been returned in the past. In networks with distributed search mechanisms, high temporal stability generally characterizes high content propagation, which may be a good indicator of limited pollution. Conversely, important levels of pollution are likely to cause low temporal stability.
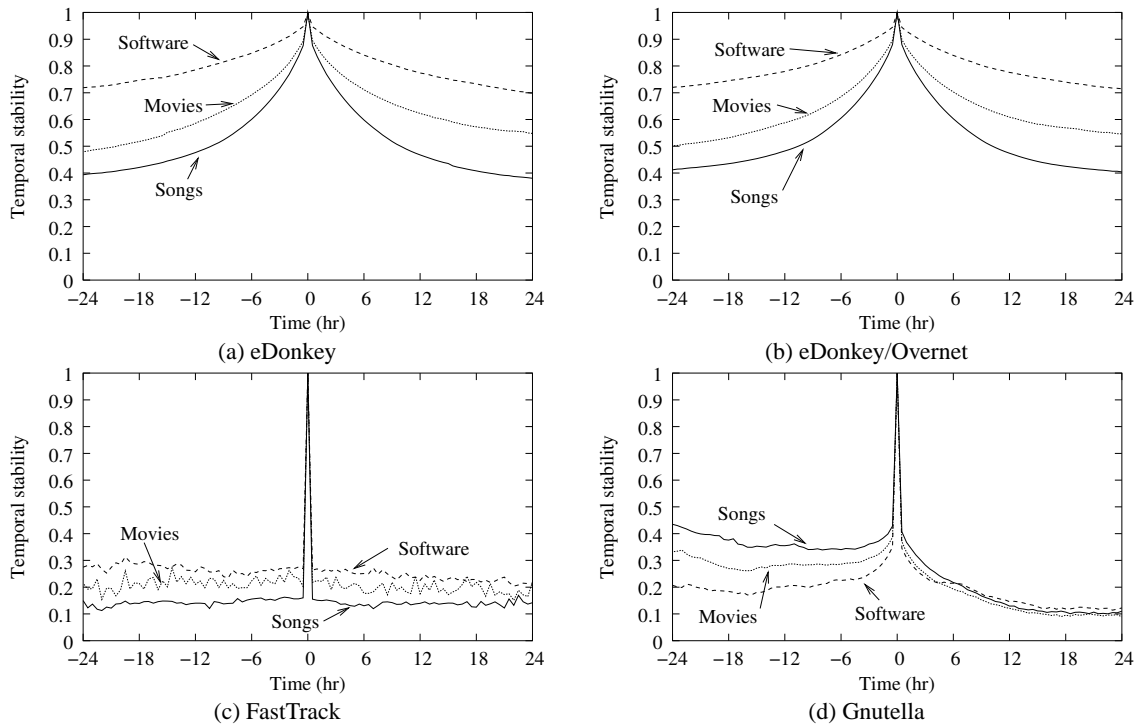
We plot the temporal stability in all four networks in Fig. 2, and observe considerable differences between the different networks. In particular, eDonkey and eDonkey/Overnet have very high temporal stability. For instance, after 24 hours, there is a 50% chance that a given user perceives a specific movie file as still being present on the network. In contrast, two factors appear to cause Fast-Track to exhibit a low temporal stability: (1) leaf-hub connections change more frequently than in eDonkey, and (2) there is a much higher pollution rate in the FastTrack network. Results for Gnu-

tella present an anomaly: judging from Fig. 2(d), content seems to be continuously disappearing from the network. In fact, we issue identical requests at a rate considered abusive by some servers, which then ban our IP addresses and stop responding to our requests. A separate experiment, whose results we omit here, shows that sending requests every hour instead of every half-hour attenuates the phenomenon.

Complementary to temporal stability, we characterize spatial stability, as a function $\sigma(n)$ of a number of hosts n. For a given query, the spatial stability is the probability that a response returned to any of the hosts is obtained, over the entire time of the experiment, by at least n different hosts. By definition, we always have $\sigma(1) = 1$.

We plot spatial stability in Fig. 3, and observe that in FastTrack and Gnutella, the probability that an item be seen at n hosts decreases exponentially in n, while eDonkey seemingly presents a more linear decrease. The exponential decrease in FastTrack and Gnutella is not surprising given the high rate of change in links between leaves and hubs, but the relatively sharp drop-off for eDonkey hints that different servers in eDonkey provide significantly different returns. Indeed, the very small number of servers in eDonkey translates into a high probability that several of our hosts are connected to the same server. Hence, we would have expected the curve to remain much closer to 1 if different servers provided relatively similar results.
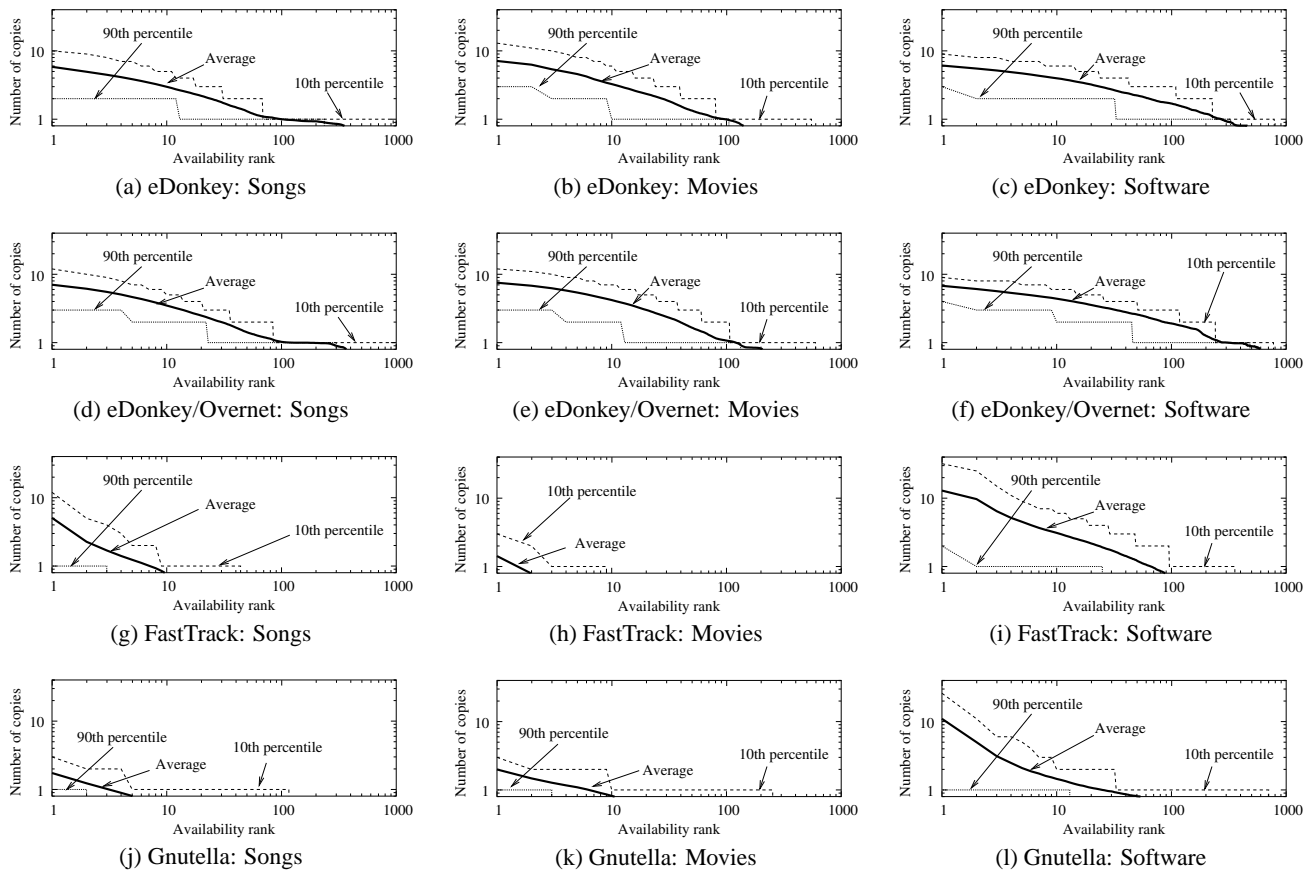
(Perceived) content replication Content replication is a direct consequence of propagation, and is perhaps the most important reason behind the success of peer-to-peer networks. Indeed, highly replicated content, being served by a number of peers, is less likely to be unavailable; in addition, most peer-to-peer protocols use swarming downloads (i.e., downloading a single file from multiple sources simultaneously), which makes replicated content easier and faster to download. Thus, most peer-to-peer clients rank query returns by number of copies of a given file found in the net-

Figure 2: Temporal stability. The plots describe the average temporal stability of the responses to all three types of query in the four networks under consideration. The temporal stability is the average probability (averaged over all times) that a response returned at a given time T is also returned at time $T + \tau$, for any τ.
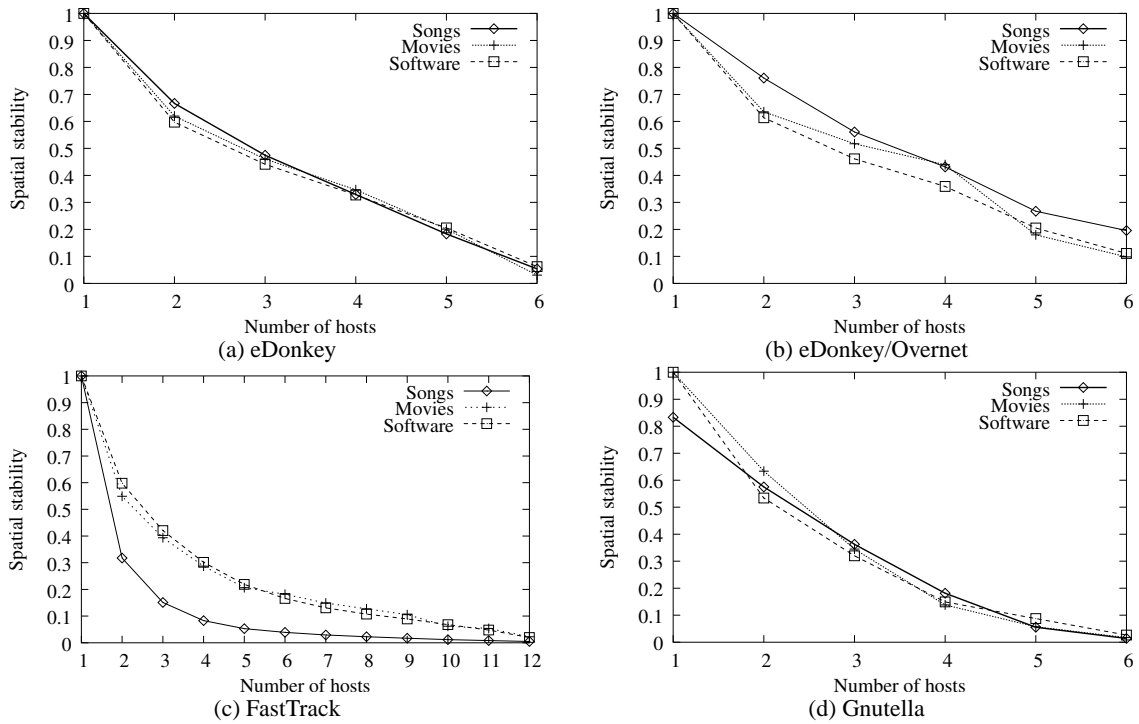
Figure 3: Spatial stability. The plots describe the average spatial stability of the responses to all three types of query in the four networks under consideration. The spatial stability is the probability that a response returned to a host is returned at least once to n different hosts, expressed in function of n.

Figure 4: (Perceived) content replication. The plots present, in a log-log scale, the average, 90th and 10th percentiles of the number of copies found against their availability rank, for all three types of queries in the four networks under consideration. Average distributions are relatively close to power-laws.

work. Ranking items according to their degree of replication creates a hysteresis effect: users are more likely to download highly replicated items, thereby increasing the number of replicas available in the network.

Fig. 4 shows that, for all networks and all items, content replication, as perceived by users, roughly follows a power law. The key result here is that we obtain the same behavior irrespective of the network considered or of the type of query; plots for the 90th and 10th percentile also exhibit power-law distributions. In addition, our results match observations previously made over entire networks for FastTrack [17], older variants of Gnutella, which did not use a two-tiered hierarchical topology, and Napster [6]. In other words, despite their limited view of the network, users have a quite accurate perception of the relative availability of different files.

Download completion time Last, we measure the total time needed to successfully complete a download. This is arguably one of the most important metrics with respect to the users' decision to abandon or join a peer-to-peer file sharing network.

Because, in this experiment, we download actual files, we use a scaled-down experimental setup to limit the aggregate amount of bandwidth we consume, and, more importantly, to only involve in the experiment machines over which we have complete administrative control.[6] We run FastTrack, eDonkey/Overnet, eDonkey, and Gnutella clients on a total six machines. Every three hours,

---

[6]Data for this experiment was collected between February 10–17, 2005.

each client sends a request for Song 1. After 10 minutes, the client ranks the query returns by number of copies found, and attempts to download the 30 highest ranked returns. Thus, our experimental setup mimics the behavior of a user who launches a query, waits long enough, and tries to download all the results she sees on her screen (most clients display about 30 results on a single screen).

We track the progress of the downloads over two hours, and plot the average number of completed downloads against time, averaged over all experiment runs, in Fig. 5. We observe that, despite its very low temporal and spatial stability, the FastTrack network is doing surprisingly well: at least one copy of the song is successfully downloaded within 20 minutes. These results indicate that, as shown in [17], the FastTrack network is highly polluted. However, good copies are easy to find among the mostly replicated objects. In other words, replication is an efficient antidote to network pollution. Gnutella shows results similar to FastTrack, albeit with lower levels of pollution, which confirms the results we previously obtained.

The eDonkey and eDonkey/Overnet clients initially lag behind the FastTrack and Gnutella clients, before catching up. This is due to the credit system used in the downloading algorithm in eDonkey. Peers which upload more traffic get more credits, and can in turn download files from a larger number of peers. Such a credit system mildly penalizes newcomers, and corroborates the results we observe. Finally, the slightly lower average of successfully completed downloads in eDonkey/Overnet compared to eDonkey does not indicate that Overnet degrades the performance of the down-
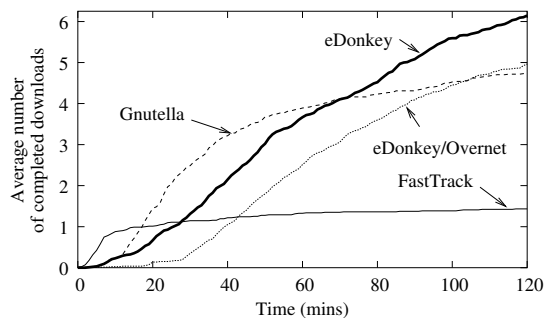
Figure 5: Download completion time. The graph shows the average number of files that were successfully downloaded after a given amount of time, averaged over all experiments.

loads, but is instead an artifact of the eDonkey/Overnet client being connected to a less efficient server in this particular experiment.

Summary of measurements   We exhibited a correlation between network topology and content availability. In particular, more centralized topologies such as eDonkey result in faster query response times, and higher temporal stability of the content available to a client. Perceived content replication, that is, the number of copies of a given file that are found in a search initiated by a node, generally follows a power law distribution. Users' perception of content replication thus matches measurements of content replication over an entire network, as reported elsewhere [6, 17]. Hence, ranking of query returns by number of replica available in the network is an effective strategy in accommodating the modest or intermediate levels of network pollution we observed in our experiments, as evidenced by the relatively short time we needed to download a good copy.

5.   POLLUTION AND POISONING

Poisoning and pollution both result in decreasing the relative availability of usable content in the network [17]. One expects pollution to be a mostly random process, which can be filtered out relatively easily as discussed in the previous section. On the other hand, poisoning should in principle be designed to ensure, with very high probability, that users always end up downloading decoys. Furthermore, we expect to observe pollution with most files in the network, whereas poisoning should be targeted to specific "protected" files.

In this section we describe three possible strategies for injecting decoys (or bad files) in a peer-to-peer network: injection of random decoys, injection of replicated decoys, and injection of replicated transient decoys. We characterize each strategy and show its effect on content availability, by considering how each strategy impacts temporal stability and number of replicas found. To that effect, we simulate each strategy on the measurements we obtained in the previous section for Song 1. We choose Song 1 because it presents typical (i.e., relatively close to the average) characteristics in the absence of poisoning, but point out that the results are mostly independent of the specific query we use in our simulations.

Random decoy injection   The first strategy we consider consists in randomly injecting decoys in the network. That is, we assume that a set of hosts advertise files that are in fact decoys, and frequently change the contents of the decoys. At low levels of injection, such a strategy is in fact a good approximation of network

pollution, if we make the assumption that polluted copies seldom propagate.[7]

At high levels, such a strategy may seem, at first glance, a rather inefficient way of poisoning an item. Indeed, flooding the network with random decoys does not, in itself, change the availability of usable files in the network. Hence, the decoys should be easy to filter out using a simple technique such as ranking search results by number of replicas found.

While this reasoning is generally true when the number of injected random decoys remains low, at high levels, we have to take into account the fact that peer-to-peer systems limit the number of returns a given query can yield. For instance, FastTrack supernodes never send more than 200 returns at a time, and can only be queried five times in a row, for a total of at most 1,000 results. Injecting a massive number of random decoys may therefore prevent usable files from appearing in the search results.

Figure 6 shows the effect of random decoy injection in the networks, for different levels of injection (0%, 25%, 50%, and 99%). We limit the number of possible query results to 2,000. We observe that while random decoy injection significantly lowers temporal stability, it does not affect content replication unless the injection level is extremely high. Fig. 6(d–f) shows that, even at high injection levels, content replication is only affected when the number of decoys injected in the network is high enough to drive usable files out of the search results.
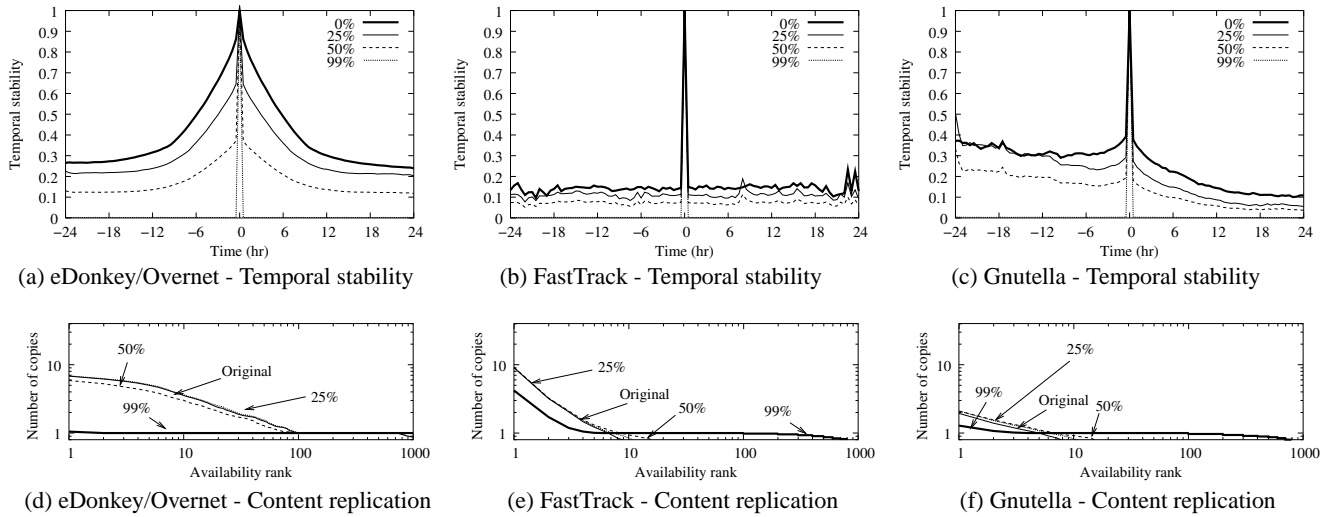
Thus, random decoy injection requires the injection of large quantities of decoys in the network to be an effective poisoning technique. For example, for an item that returns on average 100 results, one would need to inject in the order of 9,900 decoys in the network. In fact, to successfully poison the item over the entire network, one might need to inject as many as 9,900 decoys *at each hub*. While not technically infeasible, the solution is likely to be expensive and to require a massive infrastructure, which may be impractical. In addition, as discussed above, such a large injection of decoys from a limited number of sources leaves a rather obvious "signature" on the temporal stability. In highly centralized networks such as eDonkey, poisoning techniques that leave a clear statistical signature should be relatively easy to detect and combat.

Replicated decoy injection   Instead, one may consider to instead inject numerous replicas of the same decoy. Such a technique has the advantage of guaranteeing a high ranking in the search results for the injected decoy, thereby leading the decoy to be frequently downloaded. Of course, the injection of a single, highly replicated decoy is very easy to detect, so that one may improve the poisoning by injecting many replicated decoys. Liang et al. report that such a technique is used for poisoning some items in the FastTrack network [17].
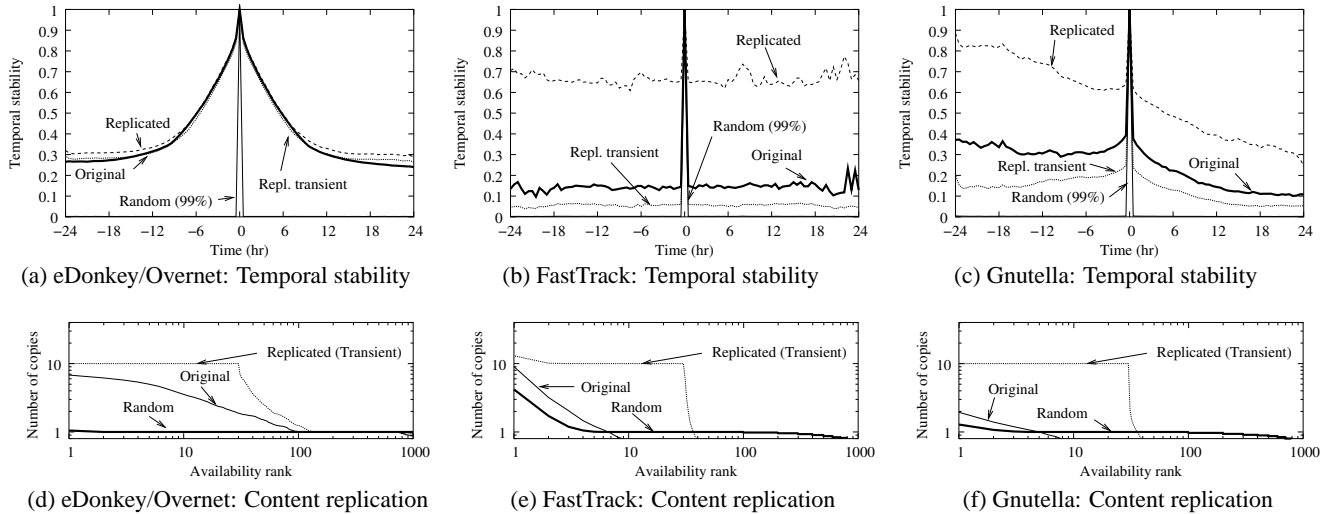
This technique is less costly that a brute-force random decoy injection. Indeed, judging from the content replication measurements we obtained in the previous section, to considerably skew the ranking of the search results in favor of the decoys, one would only need to inject about 10 replicas per decoy, and about 30–40 decoys, for a total of 300–400 files per hub . However, such a poisoning attack can be easily countered by a simple reputation system, external to the peer-to-peer network, that tells users if a given file is likely to be a decoy; the Jugle eDonkey FakeCheck service [3] is an example of such a reputation service. One can in turn defeat the reputation system by either compromising it (which may not be easy), or by frequently replacing the replicated decoys injected in the net-

---

[7]Even though this assumption is unlikely to perfectly hold, propagation of polluted files should be relatively limited, as we expect most users would delete the file once they realize it is unusable.

(a) eDonkey/Overnet - Temporal stability

(b) FastTrack - Temporal stability

(c) Gnutella - Temporal stability

(d) eDonkey/Overnet - Content replication

(e) FastTrack - Content replication

(f) Gnutella - Content replication

Figure 6: Random decoy injection. The plots describe the effect of randomly injecting decoys on the temporal stability and content replication of each network, for various levels ranging from mild pollution to aggressive poisoning.

(a) eDonkey/Overnet: Temporal stability

(b) FastTrack: Temporal stability

(c) Gnutella: Temporal stability

(d) eDonkey/Overnet: Content replication

(e) FastTrack: Content replication

(f) Gnutella: Content replication

Figure 7: Poisoning effects. The plots compare the different poisoning techniques with respect to temporal stability and perceived content replication. Replicated decoys and replicated transient decoys have identical effects on perceived content replication.

work. We call the poisoning technique of frequently replacing the replicated decoys injected in the network *replicated transient decoy injection*.

We compare the effect of the three poisoning techniques we discussed in Fig. 7: 99% random decoy injection completely destroys temporal stability, and the perception the clients have from content replication. Replicated, and replicated transient decoy injection manage to substantially skew the perceived content replication; in addition, replicated transient decoy injection avoids significantly changing the temporal stability properties of each network, and may not leave an obvious statistical signature, which makes such a poisoning strategy hard to detect.[8] In fact, except for deliberately including misspellings in metadata information to escape poisoning, we are not aware of any currently deployed mechanism that would thwart replicated transient decoy injection. A more elaborate reputation system, which weighs the reputation of a file by the

time it has been present in the network could be useful in limiting the impact of poisoning by replicated transient decoy injection.

Last, we note that the above poisoning techniques are not mutually exclusive. A poisoning attack on a file, that, for instance, combines injection of random decoys at a level of 80%, with the injection of a few replicated transient decoys, would likely be difficult to detect, and would likely lead to drastically decreasing the content availability of the targeted file.

6. CONCLUSIONS

We provided a measurement-based analysis of content availability in peer-to-peer networks. We showed that the topology of the peer-to-peer network plays a crucial role in how each peer perceives the network. Specifically, we defined the notion of temporal stability, and exhibited that more centralized topologies, such as used by eDonkey, generally have a better temporal stability than more distributed networks, such as FastTrack or Gnutella. In addition, we confirmed that centralized topologies tend to return query results faster.

---

[8]The degree of replication of the decoys, which, for simplicity, we assumed constant here, can be easily chosen to follow a power law distribution, further concealing an ongoing attack.

We showed that content replication as perceived by end users generally follows power-laws. Consequently, ranking query results by the number of copies found in the network is effective in dealing with moderate to intermediate levels of network pollution. We also discussed possible strategies that copyright holders may use to prevent the propagation of copyrighted material, and, notably item poisoning. We indicated that, to be an effective technique for reducing the availability of content on the network, randomly injecting decoys of popular files needs needs to be done on a massive scale and may be easy to detect in highly centralized peer-to-peer networks. On the other hand, the injection of a few replicated decoys can lead to significant perturbations in the network as well, while being much more cost-efficient.

We point out that more elaborate techniques, such as discussed in [10], can theoretically bring an entire peer-to-peer network down. However, copyright holders may be reluctant to disrupt an entire network and provide content protection "for free" to their competitors. Hence, whether such techniques will actually be deployed remains an open problem.

We see two main avenues for future work on the subject. First, we are interested in precisely determining the statistical characteristics of network pollution. In this paper, we reduced pollution to a random injection of bad files. However, studies of user behavior show that a vast number of users are vastly unaware of the files they share [11]. As such, one would expect polluted items to accidentally propagate, which we could determine by deliberately injecting bad copies of a file and tracking their progress in the network. Second, we focused on the network metrics which, we believe, play an important role in user behavior with respect to peer-to-peer file sharing usage. We plan on conducting laboratory experiments with human subjects to get a better characterization of user behavior in face of pollution and poisoning.

Peer-to-peer file sharing is a reality, and copyright holders seem to have, at least partially, accepted the limitations of legal recourses. Hence, we are starting to observe a technological arms race between peer-to-peer network designers and copyright holders. The former want to make their networks as robust as possible, that is, as immune as possible to poisoning and pollution, while the latter want to disrupt availability of copyrighted contents. This is one of the "tussles" Clark et al. were envisioning in [8], one which we believe will have deep economic impact in the years to come.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] A&M Records et al. v. Napster. U.S. Ct. of Appeals for the 9th Circuit, Case Nr.: 00-16401. Feb. 12, 2001.

[2] giFT: Internet File Transfer - FastTrack plug-in. http://gift-fasttrack.berlios.de/.

[3] Jugle real-time fake check for eMule and eDonkey. http://www.jugle.net.

[4] MLDonkey, a multi-networks file-sharing client. http://savannah.nongnu.org/projects/mldonkey/.

[5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. *Proc. IPTPS'03*, pp. 256–267, Berkeley, CA, Feb. 2003.

[6] J. Chu, K. Labonte, and B. Levine. Availability and locality measurements of peer-to-peer filesystems. *Proc. SPIE*, vol. 4868, pp. 310–321, Boston, MA, July 2002.

[7] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM Comp. Comm. Rev.*, 33(3):3–12, July 2003.

[8] D. Clark, J. Wroclawski, K. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's Internet. *Proc. ACM SIGCOMM'02*, pp. 347–356, Pittsburgh, PA, Aug. 2002.

[9] B. Cohen. Incentives build robustness in BitTorrent. *Proc. 1st Work. Econ. Peer-to-Peer Syst.*, Berkeley, CA, June 2003.

[10] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. *Proc. ACM SIGMETRICS'05*, Banff, AB, Canada, June 2005. To appear.

[11] N. Good and A. Krekelberg. Usability and privacy: a study of KaZaA P2P file-sharing. *Proc. ACM CHI'03*, pp. 137–144, Fort Lauderdale, FL, Apr. 2003.

[12] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. *Proc. ACM SOSP'03*, pp. 314–329, Bolton Landing, NY, Oct. 2003.

[13] J. Hale and G. Manes. Method to inhibit the identification and retrieval of proprietary media via automated search engines utilized in association with computer compatible communications network, May 2004. U.S. Patent nr. 6,732,180.

[14] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos. Is P2P dying or just hiding? *Proc. IEEE Globecom'04*, Dallas, TX, Nov. 2004.

[15] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulié. Clustering in peer-to-peer filesharing workloads. *Proc. IPTPS'04*, pp. 217–226, San Diego, CA, Feb. 2004.

[16] J. Liang, R. Kumar, and K. Ross. The KaZaA overlay: a measurement study. Working paper, Sept. 2004.

[17] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in P2P file sharing systems. *Proc. IEEE INFOCOM'05*, Miami, FL, Mar. 2005. To appear.

[18] B.-T. Loo, R. Huebsch, I. Stoica, and J. Hellerstein. The case for a hybrid P2P search infrastructure. *Proc. IPTPS'04*, pp. 141–150, San Diego, CA, Feb. 2004.

[19] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. *Proc. IPTPS'02*, pp. 53–65, Cambridge, MA, Feb. 2002.

[20] F. Oberholzer and K. Strump. The effect of file sharing on record sales: an empirical analysis. Working Paper, Mar. 2004.

[21] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of Internet content delivery systems. *Proc. USENIX OSDI'02*, pp. 156–170, Boston, MA, Dec. 2002.

[22] S. Saroiu, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. *Proc. SPIE/ACM MMCN'02*, pp. 156–170, San Jose, CA, Jan. 2002.

[23] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *Proc. ACM IMW'02*, pp. 137–150, Marseille, France, Nov. 2002.

[24] K. Tutschku. A measurement-based traffic profile of the eDonkey filesharing service. *Proc. PAM'04*, pp. 12–21, Juan-les-Pins, France, Apr. 2004.

[25] A. Zentner. Measuring the effect of music downloads on music sales. Working Paper. June 2003.