Crawling BitTorrent DHTs for Fun and Profit

Scott Wolchok and J. Alex Halderman

The University of Michigan
{swolchok, jhalderm}@eecs.umich.edu

Abstract

This paper presents two kinds of attacks based on crawling the DHTs used for distributed BitTorrent tracking. First, we show how pirates can use crawling to rebuild BitTorrent search engines just a few hours after they are shut down (*crawling for fun*). Second, we show how content owners can use related techniques to monitor pirates' behavior in preparation for legal attacks and negate any perceived anonymity of the decentralized BitTorrent architecture (*crawling for profit*).

We validate these attacks and measure their performance with a crawler we developed for the Vuze DHT. We find that we can establish a search engine with over one million torrents in under two hours using a single desktop PC. We also track 7.9 million IP addresses downloading 1.5 million torrents over 16 days. These results imply that shifting from centralized BitTorrent tracking to DHT-based tracking will have mixed results for the file sharing arms race. While it will likely make illicit torrents harder to quash, it will not help users hide their activities.

1 Introduction

The BitTorrent protocol is used to distribute a wide range of content to millions of people. BitTorrent's core design uses centralized but diverse "trackers" to coordinate peers sharing sets of files, referred to as "torrents." Users discover content out of band, often through Web-based torrent search engines and indexes that we will refer to as "torrent discovery sites."

Both trackers and torrent discovery sites have come under mounting legal pressure from major content owners [6, 7, 14, 19, 23, 25], who have in the past successfully shuttered popular services such as suprnova.org [5], OiNK [16], and TorrentSpy [18]. BitTorrent *users* also face surveillance by content owners, who are increasingly making legal threats against individuals [8, 9]. Recent work has shown that large-scale monitoring of both the major torrent discovery sites and the major trackers [1,29] is feasible. Though a boon to copyright enforcement, this capability reduces the privacy of BitTorrent downloaders by making it much harder to "hide in the crowd."

The BitTorrent community has responded to these trends by deploying decentralizing tracking systems based on distributed hash tables (DHTs). So far, their impact on privacy and copyright enforcement has been unclear. Distributed tracking is harder to disrupt through legal threats, but users still rely on centralized torrent discovery sites that remain relatively easy targets. Some might expect that decentralizing the tracking architecture would enhance privacy by making monitoring more difficult, but the question has received little research attention to date.

In this work, we demonstrate how deployed distributed tracking systems can be exploited by both sides in the BitTorrent arms race. Each can attack the other by employing recent techniques for rapidly crawling the DHTs.

On one hand, we find that DHT crawling can be used to bootstrap torrent discovery sites quickly, allowing new sites to spring up almost immediately when existing ones become inaccessible. Using only a single machine, we can create a BitTorrent search engine indexing over one million torrents in under two hours with this technique.

On the other hand, we show that it is possible to efficiently monitor BitTorrent user activity by crawling only the DHTs and *not* the centralized tracker infrastructure or torrent discovery sites. In 16 days of monitoring, we were able to observe 1.5 million torrents downloaded by 7.9 million IP addresses. Effectively, this lets us measure what content each user is sharing.

We validate our monitoring and search results by comparing them to results from previous work by other groups [1, 29] that measured the centralized BitTorrent infrastructure. In addition, we categorize our sample of torrents and confirm that the distribution is similar to previous categorizations of BitTorrent content.

The remainder of this paper is organized as follows. Section 2 introduces BitTorrent, distributed tracking, and our previous work on crawling DHTs. Section 3 explains our technique for bootstrapping BitTorrent search engines. Section 4 presents our method for profiling BitTorrent downloaders. Section 5 describes our experimental setup, Section 6 evaluates the performance of our rapid-bootstrapped search engine, and Section 7 presents the results of our user and content profiling. Section 8 surveys related work, and Section 9 concludes.

2 Background

This section provides overviews of BitTorrent, distributed tracking, and our DHT crawling methods, with a focus on details that will be relevant in our attacks.

The BitTorrent Protocol BitTorrent [2] is a peer-to-peer file sharing protocol that makes up a substantial proportion of traffic on today's Internet. One recent study [15] found that it comprised a majority (50% to 70%) of traffic in several geographic locations. The key innovation of BitTorrent is its ability to use peers' often otherwise unused upload bandwidth to share pieces of files among peers before the entire "torrent," which is the term used to refer to both a set of files being shared and the set of peers downloading and sharing those files, has finished downloading. Torrent metadata, such as the list of files contained in the torrent and checksums for each chunk or "piece" of each file, are contained in `.torrent` files, which users usually obtain from torrent discovery sites or other websites.

BitTorrent Tracking The original BitTorrent protocol relies on centralized servers known as "trackers" to inform peers about each other. To join a torrent, a peer *P* announces itself to the tracker listed in the `.torrent` file. The tracker records *P*'s presence and replies with a partial list of the peers currently downloading the torrent. *P* then connects to these peers using the BitTorrent peer protocol and begins exchanging pieces with them.

In response to legal pressure from content owners, BitTorrent developers have created extension protocols to locate and track peers in a distributed fashion, rather than relying on legally-vulnerable central servers. For our purposes, the principal distributed tracking methods of interest are the BitTorrent distributed hash tables (DHTs).

Distributed Hash Tables A distributed hash table is a decentralized key-value store that is implemented as a peer-to-peer network. Nodes and content have numeric identifiers in the same key space, and each node is responsible for hosting a replica of the content it is close to in the space. The principal DHT design used in file sharing applications today is Kademlia [21]. Kademlia nodes communicate over UDP using simple RPCs, including: STORE, which stores a set of key-value pairs at the receiving node; FIND-NODE, which requests the *k* closest contacts to a given ID from the receiver's routing table; and FIND-VALUE, which causes the receiver to return any values it is storing for a requested key. Nodes periodically replicate the values they store to the *k* peers closest to each key. The parameter *k* is specific to each Kademlia implementation.

BitTorrent has two separate Kademlia implementations: the Mainline DHT [20] and the Vuze DHT [26]. Several BitTorrent clients support the Mainline DHT, which is the larger of the two. The Vuze DHT was developed separately for the popular Vuze BitTorrent client, and typically contains about 25–50% as many nodes as Mainline. We focus on the Vuze DHT, as Section 3 explains, but our results could be extended to the Mainline DHT.

Distributed Tracking Both BitTorrent DHTs use the same general design for distributed tracking: rather than finding other peers using a tracker, a peer joins a torrent by looking up the hash of the "info" section of the `.torrent` file (known as the "infohash") as a key in the DHT and using peers from the associated value, which is simply a list of members of the torrent with that infohash. The announcement to the tracker is likewise replaced with a STORE(*infohash*, *peer_address*) operation.

In order to support fully distributed tracking, the BitTorrent community is converging [13] on a method for describing DHT-tracked torrents with a single hyperlink known as a "magnet link" instead of a traditional `.torrent` file. A magnet link contains a torrent's infohash together with minimal optional metadata, but, to keep the link short, it omits other metadata items that are necessary for downloading the files and checking their integrity. To solve this problem, BitTorrent developers have implemented several metadata exchange protocols that allow peers to query each other for a `.torrent` given its infohash. Centralized services such as torrage.com and torcache.com also allow users to download `.torrent` files given their infohashes.

BitTorrent DHT Crawler In previous work [27], we developed a crawler that rapidly captures the contents of the Vuze DHT. (We employed it in a context unrelated to BitTorrent: attacking a self-destructing data system called Vanish [11].) We briefly describe it here, referring interested readers to our earlier paper for the full details.

Our crawler, ClearView, is a C-based reimplementation of the Vuze DHT protocol. It harvests content by means of a Sybil attack [3], inserting many clients into the DHT and waiting for content to be replicated to them by nearby peers. It can efficiently host thousands of DHT clients in a single process. After most nearby content has been replicated, the clients "hop" to new locations in the DHT address space, allowing ClearView to capture more data with fewer resources.

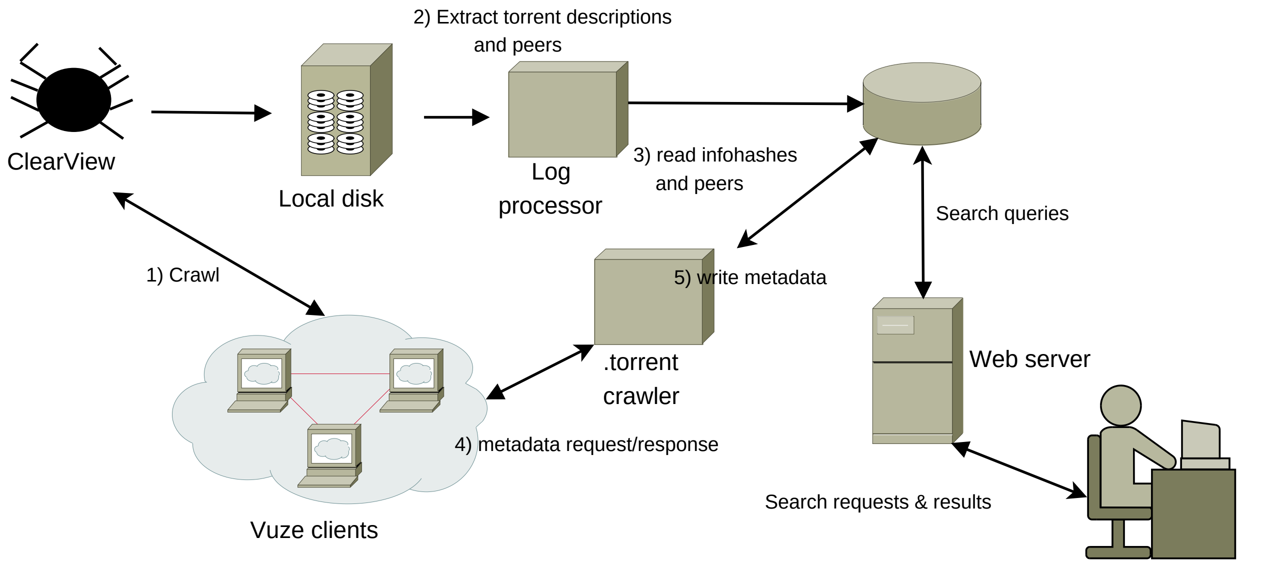
ClearView
Local disk
2) Extract torrent descriptions and peers
Log processor
3) read infohashes and peers
Search queries
1) Crawl
5) write metadata
.torrent crawler
Web server
4) metadata request/response
Search requests & results
Vuze clients

Figure 1: **SuperSeed Architecture** — Our system rapidly bootstraps a BitTorrent search engine using data from the Vuze DHT. First it surveys the DHT using our ClearView crawler and logs torrent descriptions and peer records stored there (*1*). Then it processes this data to build a search index (*2–5*), which users query through a browser-based interface.

Originally, ClearView relied on the replicate-on-join behavior of the Vuze DHT, so it only waited 3 minutes between hops. After we developed ClearView, Vuze removed the replicate-on-join behavior [10] to provide better security for Vanish. However, key-value pairs are still replicated every 30 minutes, so for this work, we configured ClearView to wait 30 minutes between hops and to use ten times as many clients per hop, yielding about the same effective coverage. Although these changes moderately increase the cost of running ClearView, our results show that crawling the Vuze DHT from a single machine is still feasible.

3 Bootstrapping BitTorrent Search

Search is a key part of the BitTorrent infrastructure. Unless users can *find* content, BitTorrent's remarkable capability to make sharing it efficient is useless. This property makes torrent discovery sites an attractive target for legal attacks from content owners, and most of today's popular search services are highly susceptible to such attacks because they are large-scale, centralized operations. DHT-based tracking has a surprising potential to change this by removing the central point-of-failure not only from tracking but from torrent discovery as well.

In this section, we describe how users can crawl BitTorrent DHTs to create and populate torrent search engines faster than they could possibly be taken down through legal attacks. This can be done using only a single machine and relying only on features already present in production BitTorrent clients. The resulting search engine could be made public, used only by its creator, or shared privately among a group of friends.

To demonstrate this approach, we built SuperSeed, a prototype rapid-bootstrapped torrent search engine. As shown in Figure 1, we use ClearView, our crawler for the Vuze DHT, to discover torrents. We then index them using whatever metadata we can learn using the DHT and associated protocols. This can include the names of the files contained in the torrent, their sizes, the peers participating in the torrent, and occasionally some free text to describe the torrent's contents.

The system handles user queries by treating the concatenation of each torrent's filenames and description as a document in the typical information retrieval model and using an inverted index to match keywords to torrents. This has the advantage of being well supported by popular open-source relational DBMSs. We rank the search results according to the popularity of the torrent, which we can infer from the number of peers listed in the DHT.

SuperSeed uses a standard crawl-index-search design. We describe each stage in detail below.

3.1 Crawling

For users to be able to download the torrents we index, we need to provide either a `.torrent` file or its infohash in the form of a magnet link (which can be used to retrieve the `.torrent` file from peers). Obtaining infohashes presented a challenge: in the Vuze DHT, peer lists are linked not to the actual torrent infohashes but to the SHA-1 hashes of the infohashes. We came across a solution when we noticed an additional class of DHT records associated with many torrents. These "torrent description" records contain the infohash along with the torrent name, its size, and counts of seeders (peers with completed downloads) and leechers (peers that have not yet downloaded the whole torrent). They are created by a prototype related-content-and-search feature in recent versions of the Vuze DHT client, which generate one for a torrent when a user downloads both it and some other torrent during the same execution of the client.

In principle, our design could also use the Mainline DHT if ClearView were extended to support it. Although the Mainline DHT does not explicitly support replication (which ClearView is designed to exploit), data must still be republished periodically, and this exposes it to crawling. We chose to demonstrate our proof of concept only on Vuze due to the significant additional complexity of supporting two DHTs in the crawler.

While Mainline does not contain torrent descriptions, its peer lists are keyed by the torrent infohashes themselves, so no additional machinery would be needed to discover infohashes. However, some method *would* be needed to discover torrent names, such as downloading `.torrent` files directly from peers using the metadata exchange protocol.

To explore this possibility, we built a prototype `.torrent` crawler. Since we are operating in the context of the Vuze DHT, our DHT crawler needs to obtain both a torrent description and a peer list before the `.torrent` crawler can contact the appropriate peers with the correct infohash. However, the `.torrent` files contain useful additional data, including the full listing of files contained in the torrent, and SuperSeed can provide them to users in place of a magnet link for BitTorrent clients that lack DHT support.

3.2 Indexing and Search

The indexing stage is largely straightforward. First, our log processor examines the crawled keys and values to identify torrent descriptions and lists of peers, which it stores in a relational DBMS (we use PostgreSQL 8.3 for our prototype). Second, we index the data and expose it to users via a browser-based interface powered by the DBMS's built-in full-text keyword search facility. For each torrent, SuperSeed's interface provides a magnet link constructed directly from the title and infohash found in its torrent description.

4 Monitoring BitTorrent Downloaders

While SuperSeed demonstrates that DHT crawling could make torrent discovery sites harder to quash, we observe that crawlers can also be used by content owners to monitor infringing activities. Although monitoring can also be accomplished by targeting the discovery sites and centralized trackers, it is easier to rate-limit or blacklist machines used for copyright enforcement in these centralized settings. The same data that we rely on for search—what content is available and from whom it can be obtained—can provide a basis for monitoring users in preparation for targeted enforcement actions.

In this section, we describe how content owners can crawl BitTorrent DHTs to monitor and track BitTorrent content and users. Like search, this can be done using only a single machine. The crawl itself is identical to that needed for search—BitTorrent DHTs cannot allow one and prevent the other.

The primary difference between monitoring and search is the length of time over which the process tracks DHT contents. For search, users are only interested in what is available *now*, so the important aspects of a BitTorrent search engine are how quickly it can be updated and what fraction of torrents it covers. In contrast, rightsholders desire to track user behavior for a specific set of content on an ongoing basis. They are interested in what content is being shared, how many users are sharing it, and who those users are.

Our monitoring design reuses the first two stages of our search design. It runs ClearView periodically and invokes SuperSeed's log processor, which stores the logs in a DBMS. It then derives content-to-IP and IP-to-content mappings from the peer lists and torrent descriptions by joining the corresponding database tables based on the infohashes. To build a longer-term picture of BitTorrent activity, the system can combine data from many crawls, just as a search engine might perform successive crawls to keep the search index up to date.

5 Experimental Setup

To evaluate our proposals for monitoring and search, we crawled the DHT for a period of 16 days from May 10 to May 25, 2010. We configured ClearView to run two 30-minute hops with 4000 Sybils each. An analytic model for the fraction of the DHT captured [27] predicts that this would record about 20% of stored values. Crawls began at 5 AM, 1 PM, and 9 PM EDT each day to match the Vuze DHT's 8-hour expiration time for stored values. The set of node IDs used by the Sybils was the same in every crawl. We report data based on 47 crawls (the 9 PM crawl on May 25 failed to complete because of an error). We began retrieving `.torrent` files at 9 PM on May 14; `.torrent` files were retrieved in a total of 33 crawls. All tests were conducted on a machine at the University of Michigan with a 4-core Intel Xeon processor, a 1 TB SATA disk running at 3 Gb/s, and 12 GB of RAM.

Class	Average (#)
Torrents with torrent descriptions	1,019,701
... and peer lists	249,371
... and `.torrent` files	56,590
Torrents with peer lists only	457,798

Table 1: **Search Coverage** — For search applications, our goal is to index a large fraction of available torrents. SuperSeed indexes torrents for which it finds a torrent description: in our experiments, more than a million torrents on average. Though obtaining `.torrent` files from peers provides useful metadata, we could do so for only a small fraction of these torrents. We can only retrieve them when our crawl data contains a torrent description and a peer list for the torrent, and even when we had both, we often could not find any responsive peers.

Component	Time (mins.)
ClearView crawl	81
Import descriptions and peers	13
`.torrent` crawl (optional)	
Get fresh peers for crawler	23
Retrieve `.torrents` from peers	30
Build inverted index	6
Total (without `.torrent` crawl)	100
Total (with `.torrent` crawl)	153

Table 2: **Rapid Bootstrapping** — In a typical run, our SuperSeed prototype bootstrapped a new torrent search site in less than 100 minutes when targeting 20% DHT coverage. An optional `.torrent` crawl to obtain extra metadata added about 53 minutes.

6 Search Evaluation

In this section, we evaluate the performance of our SuperSeed search engine prototype based on data from our crawls. The most important metrics are the resulting coverage (how many torrents we discover and index in each crawl) and the time it takes to bootstrap the search engine.

6.1 Search Coverage

The data collected in our crawls is summarized in Table 1. As we described in Section 3.2, we can only index torrents for which we recover a torrent description. Thus, SuperSeed indexed 1,019,701 torrents on average. By comparison, The Pirate Bay reported that its index contained 2.8 million torrents shortly after our experiments, and a crawl by Zhang et al. [29] counted 4.6 million torrents on five popular torrent discovery sites. (Although we did not measure what fraction of torrents could be downloaded, a search of the Pirate Bay indicates that it also indexes torrents that cannot be downloaded, such as those with zero seeders.) Each crawl was expected to recover about 20% of the values in the DHT, so this level of coverage appears quite satisfactory.

Ideally, we would like to measure what fraction of *all* available torrents were indexed, but determining the true number of available torrents is not straightforward. One way to estimate it is based on peer lists, which are spread uniformly throughout the keyspace. We recovered about 1.5 million peer lists on average, so, assuming that our crawler observed 20% of the values in the DHT, we estimate that there are 1.5 million $\times 5 = 7.5$ million torrents in the Vuze DHT on average. Based on this estimate, our index covered an average of 13% of available torrents. While this fraction may appear small, it is likely that we cover a much larger portion of *popular* torrents, since the number of torrent descriptions for each torrent is roughly proportional to its recent downloads.

While peer lists are not strictly necessary for SuperSeed's operation on the Vuze DHT, we can combine them with torrent descriptions to retrieve `.torrent` files, which provide useful additional metadata not contained in torrent descriptions. As Table 1 shows, we found an average of nearly 1.5 million peer lists, and we also found the corresponding torrent description for 249,371 torrents on average. We were able to recover `.torrent` files for an average of 56,590 (22.6%) of these torrents. Peer lists and torrent descriptions are stored under unrelated keys, so we could increase the overlap between these sets by using a larger crawl to capture a greater fraction of the DHT, or by performing a DHT LOOKUP to retrieve the peer list for each torrent description we found.

6.2 Bootstrapping Time

The startup time of SuperSeed depends on the size, content, and behavior of the DHT, which may vary for each run of the crawler, as well as the target level of coverage. Table 2 reports times for a representative execution, which took about 2.5 hours in total for a 20% DHT coverage target. As the table shows, the two key costs are crawling the DHT and importing the data from disk into PostgreSQL. The time spent to crawl the DHT is configurable and includes 21 minutes to enumerate nodes for bootstrapping ClearView. Network bandwidth permitting, the crawl time could be configured to be as low as in 35 to 45 minutes while maintaining the current coverage.

7 Monitoring Evaluation

Content owners are likely to be interested in two broad questions: (1) What content is being shared? and (2) Who is sharing it? This section evaluates how well our DHT monitoring approach can provide answers.

7.1 Monitoring Coverage

For monitoring, we define coverage as the observed fraction of torrent activity during the monitoring interval. Monitoring coverage is summarized in Table 3. We found 15,113,700 peer lists that contained 10,354,067 unique IP addresses and 14,756,483 unique IP:port combinations. Over the entire period, we recovered both peer lists and torrent descriptions for 1,538,522 torrents, to which we were able to map 7,934,945 IP addresses (11,486,776 IP:port combinations) in 45,762,064 IP-torrent pairs. This last figure can be roughly thought of as the number of observed downloads.

There were 13,575,178 peer lists without corresponding torrent descriptions; these can be archived in case torrent descriptions are recovered at a later date but are of no immediate use. We also recovered 2,046,794 torrent descriptions without corresponding peer lists; these can be used to provide a more complete picture of available BitTorrent content but have no direct evidence about the described torrents' downloaders.

The overall `.torrent` retrieval coverage was disappointing; we recovered `.torrent` files for 400,908 (26.1%) of the 1,538,522 torrents for which we discovered both a torrent description and a peer list. However, further analysis found that the 1,137,614 missed torrents were extremely likely to have few peers: 343,983 had only one peer, 514,096 had one or two peers, 622,505 had no more than three peers, and 928,315 had ten peers or fewer. Another way to see this result is to weight torrents by their peer counts: our `.torrent` downloader retrieved `.torrent` files corresponding to 37,682,371 (82%) of the 45,762,064 IP-torrent pairs for which ClearView recovered both a torrent description and a peer list.

7.2 Characterizing Content

To illustrate how this monitoring facility might be used, we characterize the available content. We begin by examining content popularity (i.e., the number of IPs associated with each torrent). As shown in Figure 2, torrent popularity follows a roughly Zipfian distribution. The most popular torrents constitute a large fraction of usage, which suggests that they should be the focus of our monitoring effort.

Next, we consider the most popular torrents. The top ten, representing less than 1% of downloads, are listed in Table 4. These are likely not the only torrents containing this content given its popularity, but they are the largest found in our crawl. Nine of the ten torrents contain video and half of them are recent episodes of television shows. We observe that all of the top ten torrents appear

Class	Total (#)
Torrents with torrent descriptions	3,585,316
... and peer lists	1,538,522
... and `.torrent` files	400,908
Torrents with peer lists only	13,575,178

Table 3: **Monitoring Coverage** — For monitoring, we want to sample activity in the DHT over an extended period of time. This table reports the total number of distinct torrents for which we recovered each kind of data, measured over all crawls. Torrents with peer lists only are a much larger fraction than in search coverage, because they tend to be unpopular and short lived.

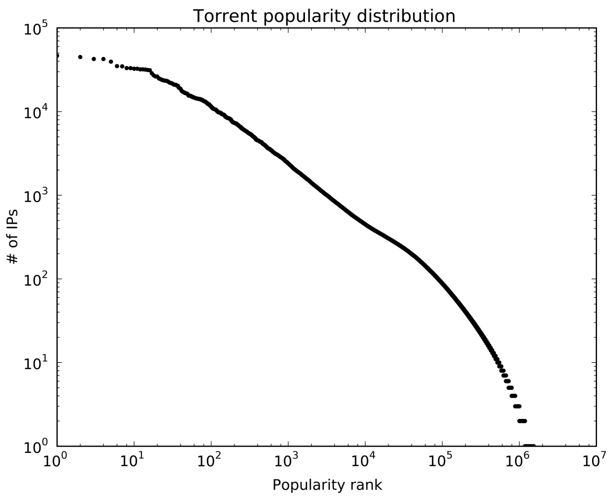

Figure 2: **Torrent Popularity** — We sorted the observed torrents by popularity and plotted torrent popularity vs. popularity rank on a log-log scale. The resulting distribution appears roughly Zipfian.

Content	Downloads
The Pacific, Part 9 (TV)	47,612
Iron Man (movie)	46,549
Alice in Wonderland (movie)	44,922
Lost, Ep. 16 (TV)	42,571
Dear John (movie)	42,562
The Back-up Plan (movie)	39,568
Lost, Ep. 13 (TV)	34,979
Desperate Housewives, Ep. 22 (TV)	34,783
The Moulin Rouge Soundtrack (music)	33,130
How I Met Your Mother, Ep. 22 (TV)	33,089

Table 4: **Top Ten Torrents** — We extracted the ten most popular torrents (by number of associated IPs) from our sample. All are apparently infringing, and half are recent episodes of television shows.

to represent illegal distribution of copyrighted content. To further investigate copyright infringement, we manually inspected the filenames of the top 1000 torrents for which we were able to download `.torrent` files, representing 13% of observed downloads. Only one file was obviously non-infringing—a Vuze data file representing a subscription to a search for “XXX” on the BtJunkie torrent discovery site—and it ranked 925th.

We used a simple rule-based classifier to categorize the complete set of torrents for which we were able to retrieve `.torrent` files. (We need the `.torrent` files to determine the filenames and extensions inside each torrent without having to download it.) The classifier examined the name and file extensions in the metadata in order to make its decision. The results of our classification are shown in Table 5. They roughly correspond to those from Zhang et al. [29] and work by Sahi and Felten [4], and match our intuitions about popular BitTorrent content (i.e., it is predominately movies, TV, and music).

Content owners may also be interested in monitoring how download activity changes over time for particular files. To demonstrate this capability, we examined changes in the popularity of the fourth most popular torrent in our dataset by considering each crawl separately. That torrent was the penultimate episode of TV’s *Lost*, which aired on Tuesday, May 18. As shown in Figure 3, the episode became available in the crawl period immediately after the one during which it aired, and its popularity peaked during our evening crawl on the Friday of the week in which it aired, suggesting that users may have used BitTorrent in part to shift their viewing of the show to the weekend.

7.3 Identifying Peers

We also studied the extent to which crawl data can be used for identifying the peers sharing each item of content. Le Blond et al. [1] observed that identifying the most active peers is challenging because BitTorrent users often hide behind proxies of various kinds in an attempt to remain anonymous. We performed a similar analysis to confirm that Vuze users in particular exhibit this behavior. Figure 4 plots the top 10,000 IP addresses by the number of torrents they joined and the number of ports they used. It is apparent that many nodes, including the four outliers in the upper right of the figure, follow a linear relation between torrents and ports, suggesting that they are proxies or NAT devices that support many users. The four outliers in particular are machines in the same /24 subnet that belong to anchorfree.com, an SSL VPN provider.

Content owners might want to distinguish proxy services from individual downloaders who access large volumes of content. Each of these two classes has a distinct, recognizable signature in Figure 4. While proxies clus-

Category	Size
Video	216,178 (53.9%)
Audio	92,128 (23.0%)
Unclassified archive	29,050 (7.2%)
Software	17,837 (4.4%)
Books	16,658 (4.2%)
Unclassified disk image	8390 (2.1%)
Games	7444 (1.9%)
Unclassified	5316 (1.3%)
Images	4216 (1.1%)
Other	3691 (0.9%)

Table 5: **Torrent Categorization** — We categorized our sample of torrents using a simple rule-based classifier.

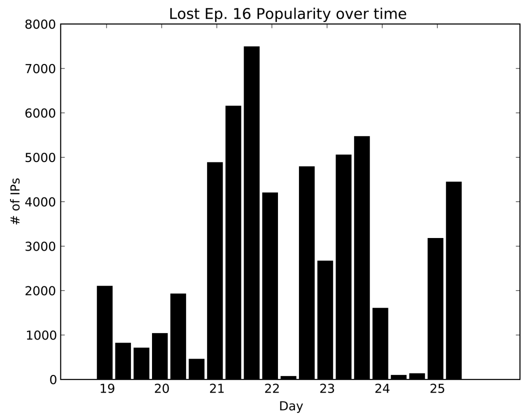

Figure 3: **Popularity Over Time** — The fourth most popular torrent in our sample was the penultimate episode of the TV show *Lost*, which aired on Tuesday, May 18th. It first appeared in our 5 AM crawl on May 19th and peaked during our 9 PM crawl on Friday, May 21st.

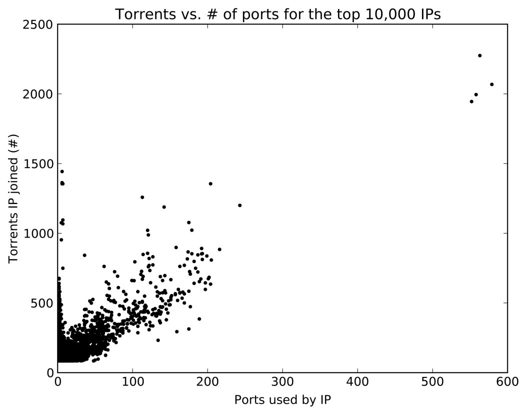

Figure 4: **Classifying IPs** — Here we plot the top 10,000 IPs by the number of torrents they joined and the number of ports they used. Outliers to the right belong to an anonymizing proxy service. Large values along the *y*-axis are likely individual heavy users or monitoring efforts.

ter around the line $y = x$ away from the origin, IPs that are likely to be individual heavy downloaders appear as points that hug the y-axis. These IPs access hundreds or even thousands of torrents on a small number of ports. (Another possibility is that some of these IPs are part of efforts to monitor torrent users by participating in the torrents. DHT-based monitoring can let us watch the watchers!)

To demonstrate our technique's ability to monitor individual users, we inspected the sets of torrents downloaded by some of the top 10,000 IP addresses that used only a single port. Individual patterns of use were clearly discernible. One user we examined was only observed downloading content that was obviously pornographic. A second user downloaded two of the top ten torrents (*Iron Man* and *The Back-up Plan*) as well as other recognizable popular content.

8 Related Work

We divide related work into two categories: searching BitTorrent content and monitoring BitTorrent at scale.

Search SuperSeed shows that BitTorrent users can respond to attacks on centralized search engines by using data in a DHT to quickly bootstrap new search sites. An alternative approach that has been widely investigated in the literature is to distribute the search functionality. For example, the usual approach to distributing search using a DHT is with an inverted index, by storing each (*keyword, list of matching documents*) pair as a key-value pair in the DHT. Joung et al. [17] describe this approach and point out its performance problems: the Zipf distribution of keywords among files results in very skewed load balance, document information is replicated once for each keyword in the document, and it is difficult to rank documents in a distributed environment. There are several proposals for improving on this basic scheme (e.g., [17,24,28]), but our approach has the advantage of relative simplicity, which is essential for the widespread adoption of a design within the greater P2P community.

Grunthal [12] concurrently developed an efficient `.torrent` indexer for the Mainline DHT. Similarly to our approach, Grunthal inserts multiple virtual nodes into the DHT, but he takes greater care not to disturb the DHT, using tens of nodes spread across multiple IP addresses rather than thousands. In contrast to our design, the nodes passively record infohashes seen in incoming FIND-VALUE requests and then actively look up the corresponding peer lists, rather than relying on incoming STOREs to deliver peer lists directly. Grunthal's crawler achieves better `.torrent` downloading performance than SuperSeed in absolute numbers, retrieving 500,000 new `.torrent` files in one week (compared to 400,000 in 16 days), although we stress that the Mainline and Vuze DHTs have different performance and usage characteristics, so a direct comparison is difficult.

Monitoring To the best of our knowledge, this is the first work to monitor BitTorrent content and downloaders at large scale using DHTs. However, much other work has focused on measuring BitTorrent at scale.

Work on large-scale monitoring by Le Blond et al. [1] is the most similar to ours. They crawled the centralized Pirate Bay tracker and associated torrent discovery site as well as the Mininova torrent discovery site, collecting 148 million IP addresses and about 2 million torrents over 103 days. Their crawler uses a single machine and is able to identify content providers and highly-active users. Although they state that they could also use the DHTs for this purpose, they provide no details to support their assertion. A major contribution of our work is that such monitoring is still possible even if one entirely ignores centralized BitTorrent infrastructure. (Indeed, the Pirate Bay tracker has since been shuttered and Mininova has been forced to filter copyrighted content, highlighting centralized infrastructure's vulnerability to attack.)

Zhang et al. [29] also crawled centralized torrent sites and trackers at large scale and obtained similar results about the feasibility of monitoring BitTorrent, although they used multiple machines. They also obtained peer lists from the Vuze and Mainline DHTs for a random sample of six torrents. Piatek, Kohno, and Krishnamurthy [22] also crawled torrent discovery sites and trackers in order to investigate content owners' monitoring of torrents.

9 Conclusion

BitTorrent's increasing usage of DHTs for decentralized tracking has significant implications for both sides of the file sharing arms race. Crawling the DHTs can be used to quickly bootstrap BitTorrent search engines as well as to monitor user activity. In our experiments, we indexed an average of over one million torrents in a single two-hour crawl, and, over the course of 16 days, we monitored nearly eight million IP addresses downloading 1.5 million torrents.

We believe that the feasibility of crawling suggests that recent legal battles over centralized torrent distribution sites are, in some sense, a distraction. Even if content owners successfully shut down these sites, BitTorrent users can easily replace them. In the long term, rightsholders may find more success by making file sharing a lower-reward, higher-risk activity through increased efforts to poison content and to take legal action against individual infringers.

Acknowledgments

The authors wish to thank Ed Felten, Kristen LeFevre, and the anonymous reviewers for valuable suggestions and feedback.

References

[1] BLOND, S. L., LEGOUT, A., LEFESSANT, F., DABBOUS, W., AND KAAFAR, M. A. Spying the world from your laptop – identifying and profiling content providers and big downloaders in BitTorrent. In *Proc. 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (Apr. 2010).

[2] COHEN, B. The BitTorrent protocol specification. http://www.bittorrent.org/beps/bep_0003.html.

[3] DOUCEUR, J. R. The Sybil attack. In *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)* (2002), pp. 251–260.

[4] FELTEN, E. Census of files available via BitTorrent. http://www.freedom-to-tinker.com/blog/felten/census-files-available-bittorrent, Jan. 2010.

[5] FISHER, K. The death of suprnova.org. http://arstechnica.com/staff/palatine/2005/12/2153.ars, Dec. 2005.

[6] FUNG, G. isoHunt sues CRIA on legality of search engines. http://isohunt.com/forum/viewtopic.php?t=141381, September 2008.

[7] FUNG, G. isoHunt sues CRIA in self defense, round 2. http://isohunt.com/forum/viewtopic.php?t=335281, November 2009.

[8] GARDNER, E. 'Hurt Locker' producer files massive antipiracy lawsuit. *The Hollywood Reporter* (May 2010).

[9] GARDNER, E. New litigation campaign quietly targets tens of thousands of movie downloaders. *The Hollywood Reporter* (Mar. 2010).

[10] GEAMBASU, R., FALKNER, J., GARDNER, P., KOHNO, T., KRISHNAMURTHY, A., AND LEVY, H. M. Experiences building security applications on DHTs. Tech. rep., University of Washington Computer Science and Engineering, September 2009.

[11] GEAMBASU, R., KOHNO, T., LEVY, A., AND LEVY, H. M. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security* (2009), pp. 299–314.

[12] GRUNTHAL, A. Efficient indexing of the BitTorrent distributed hash table. Unpublished manuscript, Esslingen University of Applied Sciences.

[13] HAZEL, G., AND NORBERG, A. BEP 9: Extension for peers to send metadata files. http://www.bittorrent.org/beps/bep_0009.html.

[14] IONESCU, D. Torrent site Mininova forced to go legit. *PCWorld* (November 2009).

[15] IPOQUE. Internet study 2008/2009. http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009.

[16] JOHNSON, B. Police shut down website after two-year music piracy inquiry. *The Guardian* (Oct. 2007).

[17] JOUNG, Y.-J., FANG, C.-T., AND YANG, L.-W. Keyword search in DHT-based peer-to-peer networks. In *Proc. 25th International Conference on Distributed Computing Systems (ICDCS)* (2005), pp. 339–348.

[18] KRAVETS, D. TorrentSpy shutters in wake of court order. http://www.wired.com/threatlevel/2008/03/torrentspy-shut/, Mar. 2008.

[19] KRAVETS, D. Torrent search engines unlawful, U.S. judge says. http://www.wired.com/threatlevel/2009/12/torrent-searchengines-unlawful/, December 2009.

[20] LOEWENSTERN, A. BEP 5: DHT protocol. http://www.bittorrent.org/beps/bep_0005.html.

[21] MAYMOUNKOV, P., AND MAZIÈRES, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS)* (2002), pp. 52–56.

[22] PIATEK, M., KOHNO, T., AND KRISHNAMURTHY, A. Challenges and directions for monitoring P2P file sharing networks -or- why my printer received a DMCA takedown notice. In *Proc. 3rd USENIX Workshop on Hot Topics in Security (HOTSEC)* (2008), pp. 1–7.

[23] RICKNÄS, M. The Pirate Bay four found guilty. *PCWorld* (April 2009).

[24] ROSENFELD, A., GOLDMAN, C. V., KAMINKA, G. A., AND KRAUS, S. An architecture for hybrid P2P free-text search. In *Proc. 11th International Workshop on Cooperative Information Agents (CIA)* (2007), Springer Berlin / Heidelberg, pp. 57–71.

[25] SAR, E. V. D. The Pirate Bay goes down following legal pressure. http://torrentfreak.com/the-pirate-bay-goes-down-following-legal-pressure-100517/, May 2010.

[26] Vuze Wiki: Distributed hash table. http://wiki.vuze.com/index.php/DHT.

[27] WOLCHOK, S., HOFFMANN, O. S., HENINGER, N., FELTEN, E. W., HALDERMAN, J. A., ROSSBACH, C. J., WATERS, B., AND WITCHEL, E. Defeating Vanish with low-cost Sybil attacks against large DHTs. In *Proc. 17th Network and Distributed System Security Symposium (NDSS)* (2010), pp. 37–51.

[28] YANG, K.-H., AND HO, J.-M. Proof: A DHT-based peer-to-peer search engine. In *Proc. 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (Dec. 2006), pp. 702–708.

[29] ZHANG, C., DHUNGEL, P., WU, D., AND ROSS, K. W. Unraveling the BitTorrent ecosystem. http://cis.poly.edu/~ross/papers/PublicEcosystem.pdf, 2009.