PPBD: A Piracy Preventing System for BT DHT Networks

Hongli Zhang, Jiantao Shi, Lin Ye
School of Computer Science and Technology
Harbin Institute of Technology
Harbin, China
Email: {zhl, shijiantao, yelin}@pact518.hit.edu.cn

Xiaojiang Du
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA, USA
Email: dxj@ieee.org

Abstract—**In this paper, we study several important issues that can be used to prevent pirated content propagation in BitTorrent (BT) Distributed Hash-Tables (DHT) networks. We design a system called PPBD to stop pirated content propagation by utilizing several attacking methods. First, the system can efficiently deal with massive concurrent connections to reduce bandwidth consumption, schedule peers to cooperate and optimize the protection methods according to clients. Second, we construct two mathematical models for BT DHT attacks, and we theoretically analyze the system performance. Third, we take into account some countermeasures of different BT clients and make corresponding optimizations of our PPBD system. Our real-world experiments show that: (1) our system can extend the download duration at least three times by the fake-block attacking method and it is more effective in a small swarm; (2) DHT index poison and routing pollution methods can limit the sharing swarm to a small swarm.**

Keywords- Peer-to-peer networking; BitTorren; DHT; piracy prevention

I. INTRODUCTION

With the ability to leverage participating users' uplink bandwidth, Peer-to-Peer (P2P) is a powerful and effective model for file sharing applications. Unfortunately, illegal users may use P2P to disseminate copyrighted materials without owners' permission. Some reports point that 75% of the piracy content occupying Internet traffic is created by BT clients [1]. Those piracy contents include movies, music, games and software, and the abuses not only hurt the financial interest of media industries, but also harm the legal use of P2P technologies. Therefore, preventing the spread of piracy content in P2P systems is an important issue. Many countries have taken administrative means to protect copyright in P2P file-sharing systems. As a result, there were shutdowns of some well-known trackers sites (e.g., BT@China_Union [2]) and manual deletions of pirated contents (e.g., Mininova [3]). These approaches do not really work well. A 'trackerless' support is added by most popular P2P applications. It uses Distributed Hash-Tables (DHT) technique to realize a fully-decentralization network and introduces "trackerless" torrent, which allows clients to use torrent files that do not have a working BT tracker [23]. Without a central component to be monitored and controlled in BT DHT, it became more difficult to prevent pirated content. Some ISPs choose providing infringement detection services to find out pirated content and warn illegal users via emails. However, most users choose to ignore such warnings. A report given in [4] shows that traffic generated by BT software resumes quickly, and the sales of the corresponding copyrighted materials drop significantly, which indicates that the piracy issues in BT networks is very serious.

From technical aspects, some new digital right management (DRM) mechanisms and copyright-protected P2P systems [9-11] have been designed. However, few of them have been deployed in practical applications. It needs a huge investment to online upgrade an Internet application system and whether users will accept the new models is unknown. Besides, some of the new systems also face the challenges of security issues such as collusion attacks. Recent years, some industries use index and content poisoning to resist illegal file sharing [5-6]. However, there is no efficient method to prevent piracy in existing BT DHT. In this paper, we design a Piracy Preventing system for BT DHT (PPBD) system. Our goal is to stop pirated file sharing propagation in the system without modifying its current architecture, nor affecting its legal users. The main idea of PPBD is trying to disrupt the process of pirate getting copyright-protected files in BT DHT systems. We insert some peers in the DHT system to intercept announcement and querying message flows of real peer indexes pointing to source peers of pirate content. We also pollute content blocks of pirate content to consequently extend the download time. PPBD system can effectively increase the probability of selecting our fake peers and decrease the probability of selecting real source peers during peer selection. The contributions of this paper are summarized below:

1) We design the PPBD system that can delay the propagation of piracy contents in BT DHT network without modifying the existing network architectures and protocols. Real experiments show that the system can limit the content sharing swarm of a pirated content to a small size and significantly increase peer download duration.

2) We construct a mathematical model for the fake-block method and a polar coordinate ID space model for the DHT pollution methods. These models are used to analyze the effectiveness and efficiency of PPBD. Our analytical results match the real-network experiments very well. Given a swarm size, the model can estimate peers' demands and bandwidth cost to achieve a target delay. The analytical results also guide the design and deployment of PPBD.

3) We optimize the protection mechanisms for popular BT clients by considering their different implementations. Our system can identify the type of a BT client and adjust the protection method accordingly. The optimization can

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

significantly improve system performance and reduce resource consumption.

II. BACKGROUD

A. BitTorrent Preliminaries

A BT system consists of four parts, including torrent index, peer index, seeds and leechers. A torrent is the meta data that stores description information of the content. A torrent index is a set of ongoing torrents that are collectively organized in the form of torrent websites. These website allow users to upload their torrents and provide tracker services. A peer index is a set of peers that participate in the distribution of a specific file. The basic function of a peer index is to track the status of peers that are currently active, and act as a rendezvous point for all peers. Depending on their download states, peers are classified into two types: seeds and leechers. A seeder is a peer that holds a complete file and uploads it to others selflessly. A leecher is a peer that has download part of a file. A leecher provides part of the file to some peers and meanwhile it downloads the rest of the file from other peers. To encourage collaboration among peers, Cohen [25] proposes a 'Tit-for-Tat' incentive mechanism to prevent selfish 'free-riding' behaviors. That scheme is implemented by the Choke/Unchoke messages. A peer gives preference to higher bandwidth peers that have uploaded to it before.

The BT protocol has acquired some new features over time, and the 'trackerless' support is the most important protocol extension [23]. DHT technique is used to decentralize the index service. Every peer is responsible to index a group of torrent. Currently almost all clients have implemented the DHT feature, and the DHT network can provide as many source peers as the tracker does. DHT network is based on the Kademlia protocol [24]. Content is identified by infohash that is stored in the torrent. Each peer has a globally unique ID (160 bits). The XOR distance is used to compare two peers or a peer and an infohash for closeness. Every peer is only in charge of torrents close to it and maintains a routing table containing some other contacts. Peers in the routing table are stored by several K-Buckets, which are organized in a binary tree. Contents are only published to the closest peers. There are four RPC messages used in the DHT protocol: Ping, Find_node, Get_peers and Announce_peer.

Figure 1 shows the flow of downloading a file in the BT DHT system. At the beginning, Peer 89 wants to download a pirated file. Peer 51 owns the file and the file's infohash is 15. After peer 51 initiates its routing table, it sends get_peers to peer 43, which is the closest node in peer 51's routing table. And then, Peer 43 responses to peer 14. Peer 14 is in the tolerance zone of the file and responses a value to Get_peers query. The value is the contact information of peer 51. At the end, peer 89 sends announce_peer to peer 14 and starts downloading from peer 51. In the next Section, we will describe how PPBD system prevents piracy propagating in the DHT network, according to the flow shown in Figure 1.

B. Utilizing Attack Methods

1) The Fake-block Attack Method. To support parallel transmissions, content in BT is divided into pieces, and each piece is further divided into blocks (usually 16). A piece corresponds to a bit in the Bitfield message. A block is the smallest unit of data transmission. After all blocks of a piece are downloaded, the client calculates a SHA-1 hash of the entire piece and compares it with the hash value stored in the torrent file. An attacker can launch the fake-block attack in which it provides a fake block. This attack causes the hash check of the piece fails, and then the client has to discard the entire piece. Our idea is to intentionally disseminate fake blocks to leechers, causing more hash check failures. This approach can waste the bandwidth of users that share piracy content and increase their download time. However, the fake-block attack method consumes bandwidth because one needs to upload fake blocks.

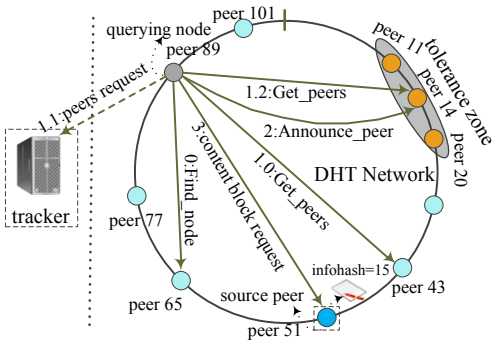

Figure 1. Content query working flow in BT DHT network

2) The DHT Index Poisoning Method. Peer index is used to help new peers bootstrap into the swarm. The index poisoning is to prevent leechers from obtaining available IP/port pairs of source peers. One approach is to poison the peer discovery mechanisms with fake IP/port pairs. A peer will have to spend a lot of time connecting to other peers. If most IP/port pairs on the peer index are poisoned. New peers can hardly bootstrap into the target swarm. However, our poison approach does not use fake IP/port but some real light weight clients. Index poisoning method increases the probability of our fake peers being connected. To perform a successful DHT index poison, we need to find out all the peer indexes that are in charge of the target torrent and poison all of them.

3) The DHT Routing Pollution Method. DHT Routing pollution aims to prevent a peer from finding correct peer indexes in the DHT network. The idea is to add many fake contacts (Sybils) into other peers' routing table. These Sybils are well designed to be close to the target file. In the DHT design, peer indexes are within a distance to the published file. This distance is called 'tolerance zone' and is identified by the first 8 bits (most significant) of the target file's infohash. If enough Sybils are inserted into the network and the routing tables of most peers in the tolerance zone are polluted, eventually Find_node query operations initiated by other peers will return our Sybils instead. Through routing pollution, attackers can control a subset of the ID space and perform other attacks (e.g., "Eclipse" attack [20] and DDos attack [16-17], etc). The search process in BT DHT is iterative, so the routing pollution method needs to pollute fewer peers than the index poison method.

C. Related Work

Recently, there have been some researches to prevent copyright infringement in P2P systems, including piracy

detecting [7-8], encryption methods for new architectures [12-13]. Few of them have been deployed in real applications, because these methods require significant changes to current software implementations. Some researchers propose using poison and pollution methods to resist piracy in BT like P2P networks. The basic idea is to reduce the availability of copyright content by disturbing the process of publishing, indexing and downloading. Yoshida *et al.* [14] apply content poisoning to one of the popular P2P file sharing applications in their local country. Wang et al. [15] proposed a copyright protection method in P2P networks by constructing false pieces with authentication collision and evaluated their work by simulations. Compared with other methods, this kind of approach requires more overhead, especially bandwidth. How to reduce the traffic cost needs further investigation. Most of the copyright content protection researches on BT system did not consider the new implementation of distributed tracker. In the DHT network there is no central component, it is more difficult to control. Although there are some proposed attacking methods in DHT, few are realized and evaluated in real world BT systems. The attacking methods including Sybil attack [19], eclipse attack [20] and routing attack [21]. Urdaneta *et al.* [18] systematically analyzed those attacking techniques. Compare to another popular DHT networks, eMule KAD [22], few work evaluated the effectiveness of these attacks in BT DHT. Since the designs of BT clients are miscellaneous, to design an effective attacking method against piracy content carrying peers is a challenge work.

III. System Design

Figure 2 illustrates the architecture of our PPBD system, which includes several components: database, TCP session manager, DHT Sybil manager and some action engines. Database is a storage container for information of pirates, such as infohash and ip/port pairs of DHT peers. If the user wants to prevent the propagation of a pirated file, a torrent infohash can be submitted to the database via the user interface. The TCP session manager handles incoming and outgoing connections and controls fake-block pollution process. The DHT Sybil manager handles incoming and outgoing UDP messages and controls poison and polluting processes over BT DHT. The two managers are front-end schedulers that dispatch work load to several backend clients. To prevent our clients from being blacklisted, the TCP session manager changes their IP/Ports periodically. The DHT Sybil manager is in charge of allocating DHT ids and IP/Port pairs to our Sybils, which conduct DHT routing pollution. We have a lot of IP/Port addresses available for the above purposes. The actual action engines include fake-block polluting engine, DHT crawler, index poison engine and routing pollution engine.

A. DHT Polluting Methods

The objectives of DHT polluting methods are:

1. Creating many lightweight 'Sybils'. These Sybils can perform fake-block polluting. Announcing IP/port pairs of these Sybils in the DHT network through broadcast. Increasing the probability of these Sybils being selected by other leechers and make fake-block pollution method more effective.

2. Creating some routing pollution peers close to the target file. Inserting these peers in the routing table of other actual peer indexes. Attract most query messages dropping into these peers. Decrease the probability of choosing other source peers and make fake-block pollution more successful.

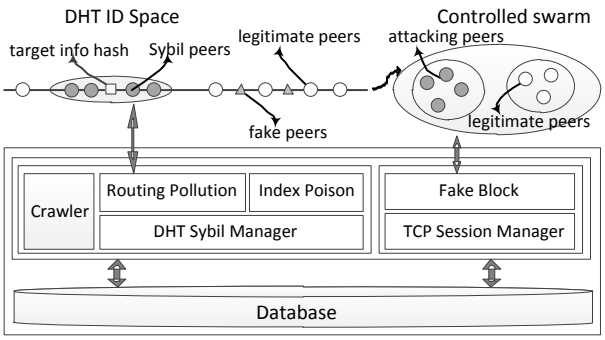

Figure 2. PPBD system architecture

In the design of PPBD, we define four important peer sets:

1. Random Peer Set: This set is created by the DHT crawler and contains tens of thousands peers. These peers cover every zone of the DHT ID space. They are the initial peer set for index poison and routing pollution procedures.

2. Critical Peer Set: This set is created through the index poison process. Peers in this set are crucial for get_peers and announce_peers queries as showed in Figure 1. These peers' routing tables contain most peer indexes of the target file.

3. Index Peer Set: This set is also created through the index poison process. Peers in the set are peer indexes of the target file and are within the tolerance zone.

4. Tolerance Peer Set: This set is created by DHT crawler. In the set, each peer ID having a 10-bit common prefix with the target infohash. The zone of tolerance peer set is larger than the zone of critical peer set.

Before causing further DHT pollutions, we need to know enough peers in the DHT networks. We use a DHT crawler to collect the information. The crawler is a modified BT client. It enters the DHT network with a random selected peer ID and its initial entries are obtained from some torrent files. The crawler sends find_node messages of random selected target key to known peers. The message flow is showed in Figure 1. We use some special designs to decrease the frequency of received UDP messages and to avoid heavy CPU usage during crawling.

Index poison: In the index poison process, we also need some crawling work to create the critical peer set and the index peer set. The message type in this stage is get_peers. The detailed algorithm is given in Procedure 1. The termination condition of the 'while' loop is no new peers being found within 15 seconds. Two types of response messages are received in this module: NODE or INDEX. The first type of message contains nodes of next hop. The crawler iteratively sends get_peers messages to these nodes. The second type of message contains information of source peers. That indicates the correspondent node is a peer index. The crawler inserts the correspondent node into the Index Peer Set and inserts its previous node to the Critical Peer Set. The last action of this procedure is to send poisoned announce_peer messages to critical peers and peer indexes. We need to poison both types of peers, because some kinds of clients do not stop searching actions after they get fake sourcing peers from a critical node, and still send find_node messages to get closer peers. A token argument needs to be included in the announce_peer message, because the peer index verifies whether the same peer has

previously sent a get_peers message. Value of the token is the same as the corresponding argument in get_peers response message. Thus, our index poison process needs to save all the received token values in get_peers response messages.

Procedure 1: IndexPoisonAttack

```
1.  get RandomPeerSet from DHT crawler;
2.  CriticalPeerSet ← Φ;
3.  IndexPeerSet ← Φ;
4.  Target ← target infohash;
5.  for p ∈ RandomPeerSet do
6.      send Get_peers(Target) Message to p;
7.  while (received response message within timeval)
8.      q ← source peer of the message
9.      type ← responded type tag;
10.     if type = NODES then // returns 8 DHT node
11.       iNode[]← received DHT nodes
12.       for i←1 to 8 do
13.           send Get_peers(Target) Message to iNode[i];
14.         iNode[i].previous ← q;
15.     else //Type = INDEX
16.       send Find_node(target) message to q;
17.       if q.previous ∉ IndexPeerSet
18.         CriticalPeerSet.insert(q.previous);
19.       IndexPeerSet.insert(q);
20. for p ∈ IndexPeerSet ∪ CriticalPeerSet do
21.     send Annouce_peer(Sybil IDs) to p;
```

Routing Pollution: DHT routing pollution attack includes active pollution method and passive pollution method. In both methods, we create several fake Sybil clients. Each Sybil client has a 140 to 110 bit distance to the target infohash. The distances are sufficient to make our Sybil clients closest to the target file.

The active pollution process starts after the crawler creates a tolerance peer set. Our fake clients send Ping messages to peers in the tolerance peer set periodically. If the victim's k-bucket in charge of the fake peer is not full, we can successfully insert the fake peer to it. If that k-bucket is full, we also have chance to insert the fake peer into the backup bucket at the same level. The backup buckets have 8 positions too. If some offline peers in real bucket are kicked off during refresh, our fake peers will promote into real bucket. After a sufficient time, the routing tables of most peers in tolerance peer set will be polluted. These fake peers will attract enough queries. Then, we can start passive pollution process.

The passive pollution gets good pollution performance by responding to external queries. All the four UDP messages are preceded in passive pollution procedure. The detailed process is given below:

1. Ping: Just copy the token field in the coming message, set our fake peers as sourcing peers and send back the Ping.

2. Find_node: When receiving find_node message, our client will check the target_id argument firstly. If the target is one of our fake peer indexes, then copy all the fields in the incoming message to the response message. Set our Sybil nodes as the closest nodes and send the response message backward. Otherwise, just discard the message

3. Get_peers: The treatment of get_peers is complex and is illustrated in Figure 3. Node P is our attacking peer contained

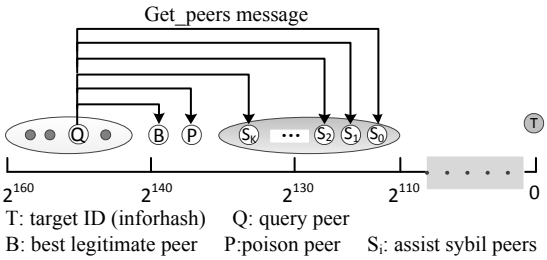

Figure 3. Working flow of treating get_peers messages during DHT index poison procedure

in the routing table of another active peer. Node Q initiates a query. When node B receives the query message, it will tell Q that P is a closer node Then Q will iteratively send get_peers to P. Since P is ours, we can control rest of the query flow. We select 8 closer peers in Sybil peers pool and reply to Q. The query is iteratively sent to those Sybils. If the Sybils are close enough, they act as peer indexes and responds get_peers messages with many forged sourcing peers. The pollution can insert many attacking peers into other active peers' routing tables and impact their peer selection procedures.

4. Announce_peers: Directly reply the messages. If the infohash is in our treatment list, send the ip/port of the source peer to fake-block pollution action engine.

Good passive polluting performance needs adequate work of active polluting. When sufficient peers are polluted by active pollution process, most of the query flows drop into our attacking peers. Many our Sybil peers insert into routing tables of other peers. In our real world experiment, we see that only several minutes after active polluting start, our passive polluting peers have received a large amount of get_peers queries. This indicates that our method is very effective.

B. Fake-block Algorithm

Procedure 2: FakeBlockAlgorithm

```
1.  if tcp connection is actively established then
2.    send Handshake Message;
3.    while true do
4.      if received handshake+Bitfield Message then
5.        Send All-1 Bitfield Message;
6.        break;
7.  if tcp connection established by other peers
        and received Handshake Message then
8.      Send handshake+All-1 Bitfiled message;
9.  while true do
10.     if received interested message then
11.       send Unchoke Message;
12.     break;
13. if received requested Message then
14.     if the requested block has been uploaded then
15.       Send Choked Message;
16.       Wait (timeval);
17.       Send Unchoked Message;
18.     else
19.       send fake block data;
```

Procedure 2 lists our fake-block polluting algorithm. At the beginning, we conduct a Sybil attack by registered many forged lightweight clients on peer indexes in the DHT network,

to attract TCP connections with other peers. Meanwhile, our client also actively initiates connections to other leechers. To let other peers accept the active connections, our clients can't act as a seed, and we only set ALL-1 bits in the Bitfield.

The clients only try to pollute one block of each piece at other peers. This is implemented differently depending on the client types. For example, if the client is the type of uTorrent, which always tries to download the whole piece from a single sourcing peer, we can extract the block number in the received request message and only pollute the first block in the piece. If the block number is not the first, we regard it as already being polluted. If the client is BitSpirit, which is prone parallel requesting within a piece, we save the recent uploading history to judge which block has been uploaded. We use choke/unchoke messages to realize the pollution.

IV. THEORETICAL ANALYSIS

A. Analysis of DHT Polluting Methods

To achieve good DHT polluting performance, we need to know that which nodes in DHT networks should be polluted. To illustrate the design ideas of our PPBD system, some theoretical analyses are given via a polar coordinate DHT ID space model, which is showed in Figure 4.

In this model, infohash of the piracy content is placed at the center of the circle. Diameter of the circle is the binary length of peer ID (160 in BT DHT). All peers locate in the circle. They can be separated to two groups by the edge of index zone. Only peers within index zone can be in charge of indexing source peers of the target content. This is because through publishing and retrieving operations, the querying peer iteratively finds the closest peers. For example, if some peer initiates a query and the closest peer in its routing table to the target is peer **c**, it starts the query to **c** for closer peers. If **c** tells that peer **b** is the closest peer. The query is sent to **b** and then iteratively sent to peer **a**. If peer **a** has no closer peer to recommend, announce_peer message is sent to peer **a**. The core of our pollution work is to find out all the peers on the edge of the index zone (e.g. peer **b**) and to insert our forged peers into these peers' routing table. In real word, there are some special implementations that make the job to be a challenge. The details are given in Section III.

Another job of DHT routing pollution method is to choose appropriate peer IDs for fake peers. If the IDs are not close enough, the control methods does not take effect. On the other hand, if the peers are too close, they are likely to be suspected by other peers. Assume the ID space contains 2^m peers in total. The length of the ID is 160. In get_peers and announcing_peer queries, 2^k closest nodes are selected as peer indexes. The size of the k-bucket is also 8. If all peers are evenly distributed in the ID space, the distance of two adjacent peers is $2^{160} / 2^m = 2^{160-m}$. Then the closest peer to the target will have m same bits prefix with the target ID. In this paper, we call the peer with a distance of 160-m bits as the target. If our fake peer has a distance closer than 160-m, our peer is closest to the target. In real world, there almost several million peers in the DHT network, that is to say m is between 20 and 24. Then the closest peer has a distance of 136 to 140 bits to the target. Hence, the best choice of our peer distance to the target is around 130 bits.

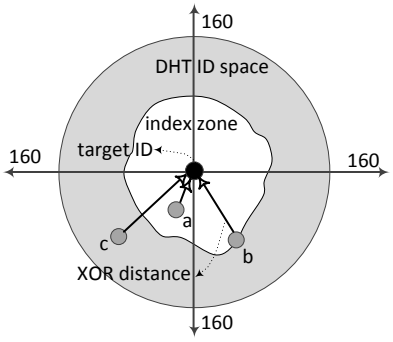

Figure 4. A polar coordinate model of BT DHT ID space

Moreover, each k-bucket of a DHT peer only has 8 positions. If the peer is very far from the target id, the k-bucket that the target ID falls into is also in charge of a large amount of IDs, then it becomes more difficult to insert a fake peer in the k-bucket. In our PPBD system, we have made some system optimization based on the above analysis.

B. Modeling of the Fake-block Method

In this subsection, we present a stochastic model for quantitative analyzing the relationship between the real impacts of fake-block method and the bandwidth consumed by the attack. We also list some important factors that impact on the pollution effectiveness. In our model, we don't consider the impact of the specific system architecture and network heterogeneity. We assume that each peer has the same bandwidth and computation capability, and peers arrive according to a Poisson process. A node leaves the swarm as soon as it finishes downloading and does not abort until it obtains the entire file. Our research focuses on the stable period in the propagation, which means that a swarm is in its equilibrium state when the number of peers is stable.

We assume that there are n peers including seeds and leechers in the swarm sharing a given file F, which is containing piracy content. F is divide into s pieces, and F_i is the i^{th} piece of F. The size of each piece is p. One piece is further divided into l blocks. Piece is the smallest content integrity-checking unit, while block is the smallest file transmission unit.. A seeder has all the s pieces, while a leecher only has a part of them. New peers arrive according to a Poisson process with a rate of λ. The downloading bandwidth of each peer is b_d. The number of sources is sufficient to allow every peer to saturate download bandwidth. After the swarm is stabilized, k fake-block polluting peers with a total upload bandwidth b_{attack}^{total} join and launch fake-block pollution method. The polluting peers claim they have all s pieces of F, and assume that they have sufficient bandwidth to accept requests from the other peers. At time t, the number of legitimate copies of piece F_i is c_i^{normal} and the number of polluted copies of piece F_i is c_i^{attack}.

We introduced an indicator to evaluate the effectiveness of fake-block pollution method on a single peer.

Definition 1 Delay Ratio: Delay Ratio is defined by $R_{delay} = t_{attack} / t_{normal}$, where t_{normal} is the download time of a peer without fake-block pollution, and t_{attack} is the download

time of a peer in the same swarm when there is a fake-block pollution. A larger value of R_{delay} indicates more serious polluting impact. If R_{delay} is very large, it means the peer can hardly finish the download.

Before calculating this indicator and analyzing the impact factors of it, we need to define two more parameters as follows:

Definition 2 (Piece Parallelism Degree): If the blocks of one piece are downloaded from ω different sources, the Piece Parallelism Degree is ω.

Definition 3 (Polluted Bandwidth Ratio): The bandwidth a peer acquired from polluting peers is denoted as b_d^{attack}, the consumed bandwidth for discarded pieces due to hash failures is denoted as $b_d^{polluted}$, the ratio $b_d^{polluted} / b_d^{attack} = \mu$ is defined as the Polluted Bandwidth Ratio.

From the assumptions of our model, we can get the following theorems.

Theorem 1. When the swarm is in its equilibrium state, the total number of normal peers in the swarm is stable. The value is given by:

$$n = \lambda \times s \times \frac{p}{b_d} \tag{1}$$

Proof. If there is no fake-block pollution in the swarm, the downloading time of a peer is given by:

$$t_{normal} = s \times \frac{p}{b_d} \tag{2}$$

Where $s \times p$ denotes the size of the file F. In the assumption of the model, new peers arrive according to a Poisson process with a rate of λ, we have $dn(t) = \lambda \times dt - n^t \times dt / t_{normal}$. When the swarm reaches its equilibrium state, it satisfies $dn(t)/dt = 0$. The number of peers at time t is $n^t = \lambda \times t_{normal}$. Using equation (2), we have Theorem 1.

Theorem 2. The value of R_{delay} is given by:

$$R_{delay} = (\frac{k + n/2}{n/2})^{\omega} \tag{3}$$

Proof. Firstly, we present some important equations:

In the equilibrium state, the number of peers that have a specific piece is given by:

$$n_1 = n_2 = \cdots = n_s = n/2 \tag{4}$$

From our assumption, new peers arrive according to a Poisson process, so the numbers of peers at different accomplishing degree are the same all the peers can be grouped to $s+1$ groups according to their accomplishing degree. The total number of piece copies is $\sum_{i=0}^{s} \frac{i \times n}{s+1} = \frac{s \times n}{2}$. The piece selection strategy in BT makes all copies from different piece evenly distributed, as shown in equation (4).

The bandwidth consumed by the pieces completely downloaded from legitimate peers is given by:

$$b_d^{normal} = p_{normal} \times b_d = (\frac{n/2}{k + n/2})^{\omega} \times b_d \tag{5}$$

In our model, all the polluting peers claim having all the pieces. Using equation (4), the probability of selecting a normal peer during peer selection is $(n/2)/(k+n/2)$, and the probability to acquire a entire piece from normal peers satisfies $p_{normal} = [(n/2)/(k+n/2)]^{\omega}$. Hence, we get the equation (5).

The download time of a peer with fake-block pollution is give by:

$$t_{attack} = s \times \frac{p}{b_d^{normal}} \tag{6}$$

Using equation (2) and equation (6), Theorem 2 can be proved.

Theorem 3. The largest value of μ is equal to l, to perform successful fake-block pollution, the attackers' bandwidth satisfies as follows:

$$b_{attack}^{total} \geq (1 - (\frac{n/2}{k + n/2})^{\omega}) \times \frac{b_d}{l} \times n \tag{7}$$

Proof. According to BT content integrity-checking method, the polluting effectiveness when there is only one block in each polluted piece is the same with the polluting effectiveness when the forged polluting clients accept all the block requests from other peers. Hence, if we only polluted one block of a piece the value of μ is equal to the number of blocks in a piece, this is the best result of all the choices.

Equation (3) indicates the fake-block polluting effectiveness. This requires that the bandwidth contributed by all the polluting peers must satisfy at least one block request of a polluted piece. The bandwidth consumed by the pieces not completely downloaded from other normal peers is $b_d^{polluted}$, it is equal to $b_d - b_d^{normal}$. The polluting bandwidth evenly allocated to each downloading peer is b_{attack}^{total} / n, only if it is larger than b_{attack}^{total} the fake-block polluting will get the best effectiveness. From equation (5), the correctness of equation (7) is proved.

Below, we list several factors that impact the fake-block pollution effectiveness.

1. The popularity of the file: Equation (7) implies that the more the downloading peers, the wider the bandwidth is required. In the equilibrium state, the total number of peers is proportional to the peers' Poisson arrival rate λ, which directly reflects the popularity of the file. Therefore, it is more difficult to obtain the best pollution effectiveness for a popular file. In the BT system, the popularity of a swarm cannot be changed. In the design of our PPBD system, we use some DHT polluting methods to prevent directly connections between other peers. The probability of selecting a valid source peer is greatly decreased.

2. The number of polluting peers: From equation (3), we can see that more polluting peers can generate a larger delay ratio, but meanwhile may decrease the upload bandwidth of each fake-block polluting peer. It is very important to select an appropriate value of k.

3. The piece parallelism degree: Equation (3) also shows that the fake-block pollution can obtain a larger delay ratio with a larger value of ω. This factor is decided by the design of the specific BT client. Some client tends to download the whole piece from one peer, while some client tends to take a parallel download strategy.

4. The polluted bandwidth ratio: A larger value μ achieves higher effectiveness. If the piece parallelism is low, it is better to upload a few fake blocks of the same piece to increase μ. Thus, the optimal choice is to only upload one block of a single piece.

V. Experimental Results

A. Experimental Environment

We set up both public and semi-public environments to evaluate our system. The public environment is mainly used by DHT experiments and consists of several servers with public IPs that can be connect directly by users from outside and the users in the same networks. Each server hosts several modified DHT clients, which can conduct passive routing pollution. The other behaviors of those clients are just like normal DHT peers, processing incoming messages and maintaining k-buckets of their local routing tables. In the public network, we also placed some lightweight clients with DHT index poison and active routing pollution behaviors. However, it is improper and illegal to directly control public torrents. In order to evaluate the effectiveness of PPBD system over a swarm of controlled torrent, we designed a semi-public environment. We placed a private tracker and several modified DHT Sybil peers in our internal networks that can be accessed locally by our own clients and cannot be connected from outside. We placed several servers with public IPs and each server hosts thousands of lightweight clients behaving as fake-block peers. These clients only registered on our local trackers. As a result, these fake-block clients can only be discovered by the BT clients in our local network. We placed another two normal BT clients in the local network to compare the difference downloading performances between fake-block pollution and non-pollution scenarios. For non-pollution scenarios, one client is installed a filtering (blacklist) function to reject connections with the fake tracker and DHT Sybil peers. The fake-block method may cause a long time to download a file. This makes the measurement inconvenient. We cannot always download the whole file. Instead, we defined an indicator to estimate the download speed. The indicator is defined by:

$$TD = \frac{T_{duration}}{D_{percentage}} \quad (8)$$

$T_{duration}$ is the download time recorded in the test and $D_{percentage}$ is the percentage of the content that has been downloaded. The equation can be used to estimate the downloading speed of the rest content that has not been finished yet.

B. Evaluation of DHT Polluting Methods

We evaluate DHT polluting methods in the public experiment environment. The clients on the experimental servers are modified versions of BT software. They don't have the capabilities to download and upload, to share pieces and to join any swarms. The client will respond to every message on the DHT serving port as a normal DHT peer, the only difference is the client will take passive routing pollution to attract queries from other peers. When receiving get_peers messages, they will respond with the registered information of peers honestly, thus they will not impact the downloading performances of other peers. We placed 20 Sybils in the network. The swarm of this torrent is estimated to including about 1000 peers. We have made several evaluations.

Attracting Announcements. To illustrate our Sybils' abilities of attracting peer announcements, every hour we use the crawler to select 20 closest peers to the target infohash. Use another client to send get_peers queries to those nodes as well as our Sybil nodes. We recorded the number of source peers carried by response messages. In order to reduce the traffic, the response message sending by our modified client only returns one UDP packet including 50 source peers. That means if there are more peers registering on our Sybil client, we will only select 50 of them. The experiment result is showed in Figure 5. At the beginning, the real peer indexes returned a lot of source peers. Three hours after our Sybils entering the network, more sourcing peers are returned by our nodes. After about ten hours, our Sybils attract most registrations. Through the experiment, our nodes did not control the entire list of peers for the target torrent even when they stayed alive for over 24 hours. There are several reasons. First, new nodes join the network all the time. We can't pollute all of their routing tables. Second, peer publishing strategies of clients are different. Some tend to publish information on the peers returned by get_peers responses, but not the closet peers in the tolerance zone. Then, some index peers will not drop into our polluting area.

Controlling Query Progress. In this experiment, we use uTorrent client in public network to perform get_peers operation to see how many our Sybil nodes are included in returned peer indexes and how many source peers are acquired from real peer indexes. Every hour, we initiated 100 queries for the target torrent, and recorded the average percentage of our Sybil nodes among all the responding peers. The result is showed in Figure 6. At the beginning, the percentage is increasing rapidly. After about 2 hours, the percentage reached about 55. After 5 hours, the percentage remained steady at between 75 and 85. The percentage kept at a high value during the rest of our experiment. We have also using a modified client to see how many sourcing peers returned by the other peer indexes. The querying results responded from our Sybil peers are filtered. Table 1 shows the results responded from real peer indexes, the returned peers are no more than 120. The longer our Sybil nodes stayed in the network, the fewer peers were returned. That is to say, even if some real peer indexes can be found, only a few source peers are returned and only a small part of them can be connected.

Polluting Routing Tables. Since there are only eight positions in each k-bucket, a Sybil peer can't always be inserted into the routing table. In this experiment, we will show the efficiency of polluting routing tables. Two DHT zones are selected and the nodes are crawled in every thirty minutes. We made a modification of the crawler to enable it record whether a Sybil peer is inserted in given peer's routing table. There are about 6000 peers in zone 1 and 3500 peers in zone 2. The result is showed in Figure 7. In the initial two hours, the polluted peers are less than 40 percent. After three hours, the percentages of polluted peers in both zones are stable and zone 2 has a better result than zone 1. The polluting is more efficient in a small zone. The polluting method can't pollute all the peers in either zone. That is mainly because peers join and leave the network all the time. Only the k-buckets of a peer stayed in the DHT for a long time will fully split and our Sybils

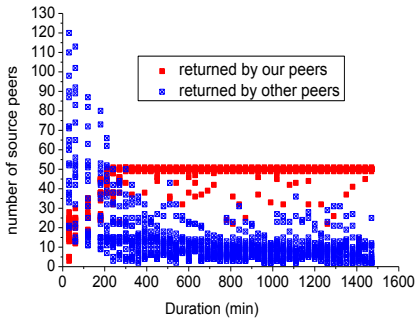

Figure 5. Returned source peers by peer indexes' response messages

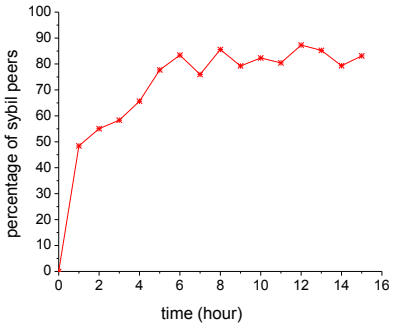

Figure 6. Percentage of PPBD Sybil peers in peer indexes' responses

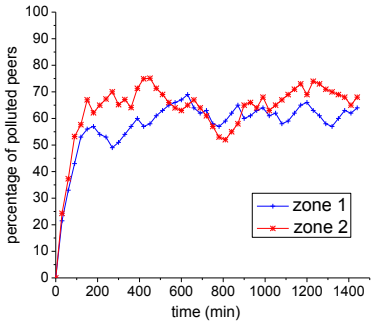

Figure 7. Effects on polluting Routing tables of peers in two DHT zones

TABLE I. NUMBER OF NORMAL AVAILABLE PEER INDEXES ACCESSED BY GET_PEERS OPERATIONS

Duration (hours)	2	4	6	16	20	24
Peer indexes	3	4	3	8	5	3
Sourcing peers	102	113	87	92	64	51

TABLE II. COMPARISON OF DOWNLOAD TIME (BITCOMET)

Name	Movie 1			
Size	196.25MB			
Scenarios	Non-attack	attack		
Ratio (fake-block / normal)	0	≈2	≈7	≈11
TD	1.93	11.6	15.17	41.56
R_{delay}	n/a.	6.01	7.86	21.53

TABLE III. COMPARISON OF DOWNLOAD TIME (BITSPIRIT)

Name	Movie 2	
Size	519.25MB	
Scenarios	Non-attack	attack
Ratio (fake-block / normal)	0	2
TD	2.02	$+\infty$
R_{delay}	n/a.	$+\infty$

will be easily inserted. Despite this, our pollution method will pollute over 60 percent of DHT peers and that is enough.

C. *Evaluation of fake-block pollution*

In this subsection, we evaluate the fake-block pollution from three aspects, the number of the fake-block polluting peers, the type of the client and the swarm size.

The Number of Polluting Peers. Experimental results are abstained from the swarms having amount of actual outside peers. Table 2 and Table 3 present the contrast of downloading time between fake-block pollution enabled scenario and non-pollution scenario of different clients types. The results show that fake-block pollution can extend the downloading time to at least six times (11.6/1.93) and bring serious performance degradation. We compared the polluting effects of different ratios between malicious peers and legitimate peers. From Table 3, we can see that the more fake-block polluting peers exist, the longer the delay will be.

Different BT Client Type. Table 2, Table 3 and Figure 9 are the results from BitComet, BitSpirit and uTorrent clients. The results indicate that, BitSpirit is more vulnerable to fake-block pollution. The main reason is that BitSpirit is prone to multiple in parallel requesting blocks of one piece from many peers, which makes the probability of getting fake blocks from polluting peers higher. The fake-block polluting toward BitComet can also have a good result even if its parallel degree is not high. uTorrent is the fastest client to finish the download among them with no more than five times duration of normal situation. It is because uTorrent uses a more progressive method to get a better downloading performance. uTorrent stops a connection if there are few activities of low traffic, and this increases the difficulties to control it. It needs more attacking bandwidth if we try to keep pollution connections, which has a side effect to make our clients more likely being blacklisted. That is why in Figure 9, the downloading speed is faster at the later phase of the experiment.

Swarm Size. Our study also shows that the factor of swarm size can influence the effects of attack behaviors. Swarm size is defined as the total number of peers sharing the same resource in BT, which indicates the popularity of a torrent. A large swarm size means that many users are interested in the corresponding torrent, and makes it harder to perform a successful pollution because a peer can connect with many other valid peers and get sufficient bandwidth. We consider two different swarms: a small swarm (697.03 MB) with around 500 seeds and 800 peers in total, and a large swarm (699.72 MB) with about 4000 seeds and 7000 peers. Figure 8 shows the results of TD using uTorrent in these two swarms. We can make the following observations. First, compared with the smaller swarm, the larger one has a short download time with a higher download rate, regardless of whether the swarm is under polluting or not. Second, our system can increase the peer download duration at least three times for both small and large swarms. Third, the larger swarm is more difficult to control. Figure 9 demonstrates the parallel download progress in normal and polluted uTorrent clients, and it shows an apparent delay in the polluted one. We have also evaluated BitSpirit to see the difference between smaller and larger swarms. In both cases, the polluted client can only obtain a small number of useful blocks in a long period of time, which means BitSpirit is more vulnerable to fake-block polluting method. As a result, DHT pollution methods can limit the swarm size and fake-block pollution has a good effect on a small swarm. That means our PPBD system can effectively prevent piracy propagation in BT DHT network.

Moreover, we have estimated the bandwidth usage of our system to control different client. The outbound traffic of successful fake-block pollution is around 100 KB/s and it is higher than inbound traffic which is around 60 KB/s, because the outbound traffic is responsible for uploading fake blocks to victim clients. BitSpirit and Vuze consume more bandwidth than uTorrent, implying that our system cannot pollute many blocks in uTorrent. uTorrent can download the resource more

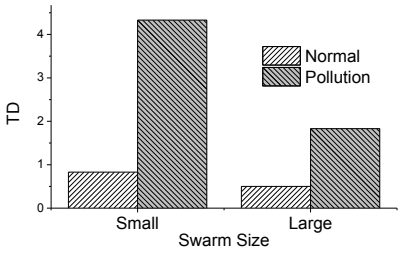

Figure 8. TD with different swarms using uTorrent

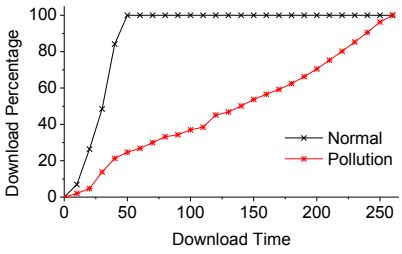

Figure 9. Parallel download progress of uTorrent

quickly as expected. The outbound traffic of DHT attack for a zone is varied at different attacking phases. The crawler will consume more bandwidth. The outbound traffic and inbound traffic are both around 200 KB/s. After the pollution is start, the inbound traffic is 40 KB/s and the outbound traffic is 100KB/s. That is because our peers attract enough query flows and the response message is larger than the query message.

VI. CONCLUSION

Piracy contents are prevailed over BT networks. It does not change the situation by shutting down tracker sites or deleting pirated content, because most popular BT clients have 'trakerless' support by equipping DHT technique. In this work, we proposed PPBD, a effective copyright protection system for BT DHT networks. The study focused on exploiting some vulnerabilities of BT system, including the fake-block method, the DHT index poison method and the routing pollution. These methods can interfere peer selection of BT clients, and delay download progress of pirates. We carried out theoretical evaluations by analyzing and modeling the behaviors of PPBD and concluded that: (1) the index poison and routing pollution methods over DHT can prevent illegal peers from finding available peer indexes; (2) the fake-block method can delay content propagation in the swarm. Furthermore, we performed real-world experiments of different swarms for existing torrents to evaluate the efficiency of PPBD, using different client types. Our experimental results show that: (1) only using the fake block method, our system can prolong the download time more than three times; (2) the index poison and routing pollution methods in DHT networks can limit the swarm to a small size. Hence, our copyright protection system is effective for most clients to prevent piracy in DHT environment.

ACKNOWLEDGMENT

This research was supported in part by the China National Basic Research Program (973 Program) under grants 2011CB302605, the China National High Technology Research and Development Program (863 Program) under grant 2010AA012504 and 2011AA010705; and by the US National Science Foundation (NSF) under grants CNS-0963578, CNS-1022552, and CNS-1065444.

REFERENCES

[1] Envisional. An Estimate of Infringing Use of the Internet. [Online]. Available: http://documents.envisional.com/docs/EnvisionalInternetUsageJan2011.pdf

[2] http://bt.btchina.net/

[3] http://www.mininova.org/

[4] DMR 2011 [Online]. Avaliable: http://www.ifpi.org/content/section_resources/dmr2011.html

[5] J.Liang et al., "The Index Poisoning Attack in P2P File-Sharing Systems", IEEE INFOCOM, 2006.

[6] J.Liang, R. Kumar, Y. Xi, K.W. Ross. "Pollution in P2P File Sharing Systems", IEEE INFOCOM 2005, Miami, March 2005

[7] J.Mee, P.A.Watters, "Detecting and Tracing Copyright Infringements in P2P Networks," ICN/ICONS/MCL 2006.

[8] K.P. Chow, K.Y. Cheng, L.Y. Man, Pierre K.Y. Lai, Lucas C.K. Hui, C.F. Chong, K.H. Pun, W.W. Tsang, H.W. Chan, S.M. Yiu, "BTM - An Automated Rule-based BT Monitoring System for Piracy Detection," . ICIMP 2007.

[9] D.Tsolis, S.Sioutas, A.Panaretos, I.Karydis, K.Oikonomou, "Decentralized digital content exchange and copyright protection via P2P networks," Computers and Communications (ISCC), 2011

[10] Q.Qiu, Z.Tang, Y.Y.Yu, "A decentralized authorization scheme for DRM in P2P file-sharing systems,".Consumer Communications and Networking Conference (CCNC), 2011

[11] X. Zhang, D. Liu, S. Chen, Z. Zhang, and R. Sandhu, "Towards Digital Rights Protection in BitTorrent-like P2P Systems," 15th SPIE/ACM Multimedia Computing and Networking (MMCN'08),

[12] Y.Y.Chen, J.K.Jan, Y.Y.Chi, M.L.Tsai, "A Feasible DRM Mechanism for BT-Like P2P System," International Symposium on Information Engineering and Electronic Commerce (IEEC), 2009

[13] X.S.Lou, K.Hwang, "Collusive Piracy Prevention in P2P Content Delivery Networks", IEEE Transactions on Computers, July 2009.

[14] M. Yoshida, S. Ohzahata, A. Nakao, and K. Kawashima, "Controlling File Distribution in The Share Network Through Content Poisoning," AINA 2010

[15] C. Wang and C. Chiu, "Copyright Protection in P2P Networks by False Pieces," international conference on Autonomic and trusted computing, 2011.

[16] K.E.Defrawy, M.Gjoka, A.Markopoulou, "BotTorrent: misusing BitTorrent to launch DDoS attacks," the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet, 2007.

[17] K.C.Sia, "DDoS Vulnerability Analysis of BitTorrent Protocol,". UCLA:Techical Report, 2006.

[18] G.Urdaneta, U.Pierre, M.V.Steen , "A survey of DHT security techniques," ACM Computing Surveys, Volume 43 Issue 2, 2011

[19] J. R .Douceur, "The Sybil attack." the 1st International Workshop on Peer-to-Peer Systems, Germany, 2002

[20] E.Sit, R.Morris, "Security considerations for peer-to-peer distributed hash tables," the 1st International Workshop on Peer-to-Peer Systems, Germany, 2002

[21] A.Singh et al. "Eclipse attacks on overlay networks: Threats and defenses," IEEE INFOCOM, 2006.

[22] M.Steiner, T.En-Najjary, E. W. Biersack, "Exploiting KAD: possible uses and misuses," ACM SIGCOMM Computer Communication Review, 2007

[23] Azureus (now called Vuze) Bittorrent Client , [Online]. Available: http://azureus.sourceforge.net/plugin_details.php?plugin=mlDHT

[24] P.Maymounkov, D.Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric.," IPTPS 2002.

[25] Cohen, "Incentives build robustness in BitTorrent," the first Workshop on Economics of Peer-to-Peer Systems, 2003.