Preventing Piracy Content Propagation in Peer-to-Peer Networks

Hongli Zhang, Lin Ye, Jiantao Shi, Xiaojiang Du, and Hsiao-Hwa Chen

***Abstract*—Peer-to-Peer (P2P) networks have been widely used in various Internet applications. However, P2P networks also cause serious concerns of copyrighted contents piracy: P2P helps to pirate copyrighted contents, which impedes wide application of P2P networks. In this paper, we investigate several important issues on the ways to prevent pirated content propagation in P2P networks. We propose a system to stop pirated content propagation by utilizing several attacks to BitTorrent (BT). First, we design a system that can handle a large number of concurrent connections to reduce bandwidth consumption. Second, we build two mathematical models for BT attacks, including leechers behavior model and fake-block behavior model, and we analyze the performance of the system using these models. Third, we optimize BT clients by taking into account different implementations and counter-measure designs. We conducted several experiments in real networks, the results of which verifi d that our system is suitable for most current BT clients, and BT clients' download duration is extended at least three times longer, which is much better than the results reported in the literature.**

***Index Terms*—Peer-to-peer networking; BitTorrent; piracy prevention.**

I. Introduction

Peer-to-Peer (P2P) networking provides an eff cient way to share resources amongst a large number of users, and they have been widely used in the Internet applications. Unfortunately, illegal entities may use P2P to disseminate copyrighted materials without owners' permission. As reported by Envisional [1], at least 23.76% of Internet traff c is piracy content, 75% of which uses BitTorrent (BT). Those contents include movies, music, games and softwares. These abuses discourage content providers and also impede the wide application of P2P technologies.

Copyright protection in P2P networks is an important issue. Many countries have passed the laws to protect digital copyrights. As a result, there were shutdowns of well-known websites (e.g., BT@China_Union [2]) and deletions of pirated contents (e.g., Mininova [3]). However, the study given in [4] showed that traffi to these websites resumes quickly, and the sales of the corresponding copyrighted materials drop significantly, indicating a continuing growth of privacy. A number of counter-measures have been proposed to protect copyrights in P2P f le-sharing systems. However, very few of them have been successfully deployed in practical applications. Recently, anti-P2P companies [5] [6] have started using some forms of attacks to prevent illegal f le sharing in the Internet. However, it is not clear how well this approach works.

A. Preliminaries of BitTorrent

A BT system consists of four parts, including torrent index, peer index, seeds, and leechers. A torrent is the meta data that store descriptive information of a file A torrent is important for the users because it contains necessary information to bootstrap users into a collection of peers (also called a swarm). A torrent index is a set of ongoing torrents that are collectively organized in the form of torrent websites, which 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 specifi file The basic function of a peer index is to track the status of peers that are currently active, and acts as a rendezvous point for all peers. There are three index schemes in BT, i.e., tracker, Distributed Hash Tables (DHTs) [7], and Peer EXchange (PEX) [8]. In the tracker scheme, each peer registers itself to a server called tracker, and each peer may also obtain a random subset of other peers (IP addresses and Port #). DHT is used to support distributed index maintenance. In DHT, a client (called a DHT node) can query "infohash" to obtain a similar result of the other peers, where "infohash" is a unique f ngerprint of a .torrent f le. PEX is an alternative way that allows the peers to exchange the information of their active neighbors with each other.

Depending on their download states, peers are classif ed into two types: seeds and leechers. A seed is a peer that has already downloaded a fil and is willing to provide the f le to other peers even though it does not need any more contents. A leecher is a peer that has downloaded part of a file A leecher provides part of a f le (that it downloaded) to the other peers, and meanwhile it downloads the rest of the f le from the other peers.

To encourage collaboration among peers, Cohen [9] proposed a "Tit-for-Tat" incentive mechanism to prevent selfis "free-riding" behaviors. In general, each peer has many simultaneous connections with the others, but it can upload part of them (default number is four). A peer gives preference to higher bandwidth peers that have uploaded to it before. In addition, a peer reallocates an upload slot to a random peer every 30 seconds, which is also called optimistic unchoking, which offers two advantages. First, it is favorable to seek for

Manuscript received March 1, 2012; revised July 14, 2012.

H. Zhang, L. Ye, and Jiantao Shi are with the Harbin Institute of Technology, Harbin, China (e-mail: zhanghongli@hit.edu.cn, {yelin, shijiantao}@pact518.hit.edu.cn).

X. Du is with the Temple University, Philadelphia, USA (e-mail: dux@temple.edu).

H.-H. Chen is with the National Cheng Kung University, Tainan City, Taiwan (e-mail: hshwchen@ieee.org).

This research was supported in parts by China National Basic Research Program (973 Program) grants 2011CB302605, China National High Technology Research and Development Program (863 Program) grants 2011AA010705 and 2012AA012506, the US National Science Foundation (NSF) grants CNS-0963578, CNS-1022552, and CNS-1065444, and Taiwan National Science Council grant NSC99-2221-E-006-016-MY3.

Digital Object Identifie 10.1109/JSAC.2013.SUP.0513010

the peers with higher download rates for better performance. Second, it allows new peers to obtain upload slots although they have not uploaded any content to the others yet.

B. Related Works

Many counter-measures for preventing copyright infringements have been proposed in P2P systems. These approaches can be divided into four categories.

1) Detecting piracy: Typically, piracy detection systems identify copyright infringements by analyzing users' traff c and behaviors, such as BT monitoring system using certain rules [10]. These systems can log infringement actions as evidences and warn the misbehaving users [11]. The work in [12] used trusted auditors to detect illegal peers during the interactions. However, deep packet inspection may invade users' privacy.

2) Designing new copyright-protected systems: Some have proposed many new copyright-protected P2P systems and digital rights management mechanisms, such as f ngerprinting [13], watermarking [14], encryption [15], etc. Typically, an authorization server assigns a unique ID or license to each paid peer, which will be served after verification An encryption scheme encrypts content pieces before transmission. Peers must pay for the keys to decrypt them. However, new architectures may bring new security issues, e.g., encryption makes the systems vulnerable to pollution attacks. Besides, very few have been deployed in real applications even though most methods were well designed. The work in [29] describes the design, implementation, and experience with OneSwarm, a new P2P data sharing system that provides users with explicit, conf gurable control over their data. OneSwarm focuses on privacy-preserving data sharing in P2P where users are not willing to expose their downloading behaviors to the public. Although OneSwarm is able to share data anonymously by designing sharing permissions and trust at the granularity of individual data objects and peers, it cannot prevent piracy content propagation in existing BitTorrent overlay.

3) Enhancing existing systems: The work in [16] suggested a piece-level encryption and re-encryption method based on ElGamal cryptosystem in BitTorrent-like systems. Each peer has its own key and needs to purchase it for decryption. The study carried out in [17] proposed a peer authorization protocol to distinguish illegal clients from legitimate ones. These improvements require additional hardware and modif cation to clients at the expense of scalability and eff ciency.

4) Preventing piracy in existing systems: The basic idea is to reduce the availability of piracy by disturbing the processes of publishing, indexing and downloading. The work in [18] exploited content poisoning to control f le distribution in shared networks. The study in [19] protected copyright contents in P2P systems by constructing false pieces with authentication collision. Compared with 2) and 3), this approach requires more overhead, especially bandwidth.

C. Challenging Issues

In this work, our focus is on the control of piracy in existing BT networks to prevent piracy by combining index and data poisoning. The vulnerabilities of a BT system have been analyzed in the literature. The experiments in [20] illustrated many peers were attacking or polluting BT systems, such as uncooperative-peer attacks and fake-block attacks. The attacks were also summarized in [21]. The work in [22] gave a detail study on lying piece possession attack and Sybil attack. The deceitful piece reports can result in an imbalance in the amount of replicas for each piece, which induces the local rarest policy to make wrong piece selections to bring in more rarest pieces. To achieve our goal, there are still several challenges to be tackled.

1) There are thousands of peers in a swarm with high-level service capacity. It is a challenge to design a copyright protection system that can control the propagation in real-world Internet. We need to deal with numerous concurrent connections and reduce resource consumption.
2) BT (e.g., "Tit-for-Tat" and piece hash check, etc.) is highly dynamic and complex that makes the modeling of attacks extremely diff cult. We need to evaluate our system to guide the deployment and predict performance.
3) There are many clients with different features. For example, blocks of a same piece are usually downloaded from multiple peers in BitSpirit [23], but these blocks come from the same peer in BitComet [24]. Furthermore, uTorrent [25] is able to identify malicious peers based on piece check failure, and Vuze [26] can identify them by a strict block-to-block check. We need to design appropriate methods to enhance system effectiveness.

D. Our Contributions

In this paper, we focus on one of the most popular P2P applications, i.e., BT. Our goal is to stop pirated file- haring without modifying the existing BT systems. The main idea of preventing piracy in our approach is to delay/disrupt the process of obtaining a f le in BT. Specif cally, we poison the index and content of a file which reduces fil availability and increases download time of a pirated f le. Our contributions can be summarized as follows.

1) We design a copyright protection system that can delay the propagation of piracy contents in BT without modifying the existing system and protocols. Our system is suitable for most current BT clients. Real experiments showed that the system can increase peer download duration by at least three times longer.
2) We build two mathematical models for BT attacks, i.e., leechers behavior model and fake-block behavior model. We use the models to analyze BT attacks. Our analytical results match to real-network experiments very well. Given a swarm size, these models can estimate peers' demands and bandwidth costs to achieve an expected delay. These results can also guide the design and deployment of the proposed copyright protection system.
3) We optimize the protection mechanisms for major BT clients by considering their different implementations. Our system can identify the type of a BT client and adjust the protection method accordingly. This optimization can significa tly improve system performance and reduce resource consumption.

The rest of the paper is outlined as follows. In Section II, we describe the design and implementation of our proposed system. In Section III, we present theoretical analyses of BT attacks. In Section IV, we discuss the real-world experiments, followed by the conclusions made for this paper in Section V.

II. System Design and Implementation

Fig. 1 illustrates the architecture of our proposed Content Protection (CP) system, which includes four components, i.e., database, BT session manager, action engine, and control plugins. Database is a storage container for information of pirates, such as tracker list and DHT nodes. If someone wants to stop or prevent the propagation of a pirated f le, a torrent can be submitted to the database via a user interface. BT session manager handles incoming and outgoing connections. The BT session manager is also a front-end scheduler to dispatch load to several backend controlled clients, and send response messages to the corresponding clients. To prevent controlled clients from being blacklisted, the BT session manager changes the IP/Port of controlled clients that send blocks to the same victim client. We have a lot of IP/Port addresses available for the above purposes. The action engine is the core of our system whose job is to parse incoming messages, identify client types, decide which method should be used to delay a download process, and assemble the messages. Each method (such as index poisoning, leechers, and fake-block attack) is plugged into the action engine. The index poisoning attack takes three forms, i.e., tracker, DHT and PEX. The tracker and DHT methods are independent to the BT session manager, so that they are located outside the diagram in Fig. 1.

A. Content Protection Utilizing Existing BT Attacks

1) Index Poisoning Attack: A peer index is the f rst step for each peer to bootstrap itself into a swarm. Some peers of a swarm may be unavailable (e.g., when they are off ine), and then a peer will have to spend a lot of time to connect to the other peers. The index poisoning attack takes advantage of this fact and poisons the peer index with a lot of fake IP/Port pairs.

In our CP system, we utilize three forms of index poisoning attacks, which are the attacks on three peer index schemes, i.e., tracker, DHT, and PEX. Tracker and DHT have been discussed in [21]. PEX is an eff cient way for a malicious client (an attacker) to poison its neighbors. After establishing connections to legitimate peers, the attacker responds to BT handshake and sends out PEX messages using fake IP/Port(s). By using PEX, the IP address of an attacker is propagated to many peers. The index poisoning attack provides a basis to launch the other advanced attacks, such as connection attack and fake-block attack.

2) Connection Attack: For performance and security reasons, BT client usually bounds the number of simultaneous active connections for each task. The basic idea of a connection attack is to set up as many connections with legitimate peers as possible, which in turn decreases the number of available connections to the other peers. To launch this attack, the attacker needs to establish a large number of connections to leechers. To maintain the connections, the attacker has to periodically exchange BT messages with the leechers.

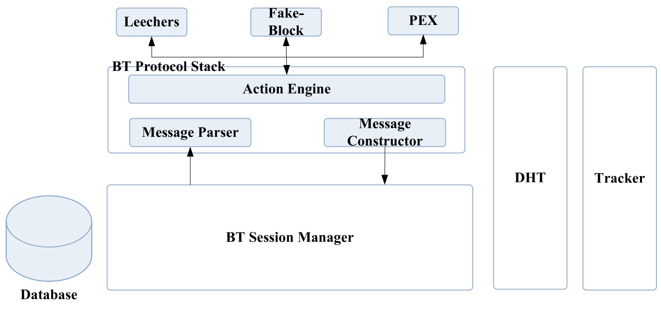

Fig. 1. The architecture of content protection system.

Our CP system utilizes one type of the connection attacks, or leechers attack, where an attacker sends piece requests to peers and downloads the corresponding pieces which generate one-way traffic Because this attack does not require the attacker to upload any block, it is considered as a “free-riding” leecher and it is choked by the other peers. After being choked, the leechers attackers try to keep the connection.

3) Fake-block Attack: To support parallel transmissions, each resource unit in BT is divided into pieces, and each piece is further divided into blocks (usually 16). A piece corresponds to a bit in a Bitfiel message. A block is the smallest unit of a successful transmission. After all blocks of a piece are downloaded, the client will calculate a SHA-1 hash of the entire piece and compare it with the hash value stored in the .torrent file An attacker can launch a fake-block attack in which it provides a fake block. This attack will cause hash check of the piece fail, and then the client will have to discard the entire piece.

Our idea is to intentionally disseminate fake blocks to leechers, causing more hash check failures. This approach will waste illegal users' bandwidth and increase their download time. However, the fake-block attack needs bandwidth to upload fake blocks. In order to reduce bandwidth requirement, our system just sends illegal users one fake block instead of the whole piece because one fake block is sufficien to cause a hash check failure.

B. Content Protection Utilizing Improved BT Attacks

1) Progressive Connection: As an open protocol, BT does not give the specif cations on how to implement a client. To have a higher market share, some developers have made their own efforts in improving BT client download performance. Progressive connection is one of these efforts. When connected with slow peers, some clients may disconnect the slow peers and try to connect to some faster ones. For example, if uTorrent [25] f nds there is no activity in a connection, the client will stop it. On the other hand, BitSpirit [23] does nothing to the slow connections.

Compared to the connection attack, the fake-block attack requires more bandwidth. Our CP system is able to identify the types of BT clients according to their PeerIDs that are generated randomly at the f rst client installation. If a client is recognized as BitSpirit, the connection attack is preferred, which usually ties up most connections of an illegal user. If a client is uTorrent, the connection attack does not work well because a uTorrent client will try to connect to the other peers. The fake-block attack works better on uTorrent clients.

2) Blacklist Mechanism: Many clients use a blacklist to block potential misbehavior peers. There are static and dynamic blacklists. A static blacklist provides a conf guration fil for defi ing f ltering rules. However, static blacklists cannot identify unknown or camouf aged peers that continually change their IP/Port addresses. Therefore, a dynamic blacklist is often used to record the source of a bad piece and determine which peers are malicious. For example, a uTorrent peer has a counter for each connecting peer. When the hash check of a piece fails, the uTorrent peer increases the counter for each peer who contributes the blocks to that piece. When the counter is over a given threshold, the peer is banned and added into a (dynamic) blacklist. Vuze does not discard a bad piece until a good piece is downloaded. In Vuze, a client compares bad and good pieces block-by-block to f nd out the peers who send fake blocks.

The fake-block attack was carefully designed to keep attacker out off the blacklist. Our CP system has a central scheduler to monitor each peer's uploading. Furthermore, we utilize a blacklist in uTorrent because a hash check failure increases the counters of both legitimate and malicious peers. This means that a legitimate peer will also be banned. In addition, the more data a legitimate peer uploads, the higher probability it will be banned.

III. Theoretical Analyses

Theoretical analyses are useful for guiding the design of our system. In this paper, we perform analyses on two attacks, i.e., leechers attack and fake-block attack, which are presented in Subsections II.A and II.B, respectively.

A. Valid Connection Model Based on Leechers Attack

Leechers attack [27] [28] is one of the most common attacks on BT systems. Leechers attack exploits BT in two aspects. First, the weakness of authentication allows an attacker to easily generate thousands of forged identities using only a small number of machines. Attackers can increase, decrease or change those identities dynamically. Second, the random selection mechanism in an optimistic unchoking algorithm [9] allows a peer to obtain connections from the others even if it never uploaded blocks. The leechers attack is illustrated in Fig. 2.

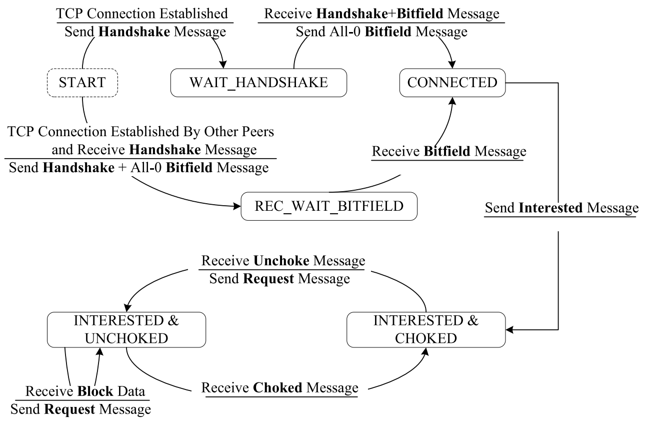
TCP Connection Established
Send **Handshake** Message
Receive **Handshake**+**Bitfield** Message
Send All-0 **Bitfield** Message
START
WAIT_HANDSHAKE
CONNECTED
TCP Connection Established By Other Peers and Receive **Handshake** Message
Send **Handshake** + All-0 **Bitfield** Message
Receive **Bitfield** Message
REC_WAIT_BITFIELD
Send **Interested** Message
Receive **Unchoke** Message
Send **Request** Message
INTERESTED & UNCHOKED
INTERESTED & CHOKED
Receive **Block** Data
Send **Request** Message
Receive **Choked** Message

Fig. 2. The state diagram of leechers attack.

First, let us analyze the damage of leechers attacks, and we use the following notations: n is the number of legitimate peers in a swarm, m is the number of upload connections of a legitimate peer, and k is the number of malicious peers (attackers) in a swarm.

A connection is considered as valid if the connection transmits actual data. The number of valid connections is $m \times n$. In normal situation, all connections are set up among legitimate peers and are valid. When a swarm is under attack, some connections are occupied by malicious peers and the bandwidths of these connections are wasted. In the text followed, we study the distribution of valid connections between legitimate and malicious peers.

Denote $F(n,m,k,t)$ as the number of valid connections obtained by a set of legitimate peers under a leechers attack after the tth round optimistic unchoking. Denote $P(n,m,k,t)$ as the ratio between the numbers of valid connections after and before an attack.

Theorem 1. The number of valid connections obtained by legitimate peers after the tth round optimistic unchoking is

$$F(n,m,k,t) = \frac{k \times n \times (m-1)^t + n^2 \times m^t}{(n+k) \times m^{(t-1)}}, \quad t > 0. \quad (1)$$

Proof. If there is no attack ($t = 0$), i.e., there is no malicious peer ($k = 0$), the number of valid connections from a set of legitimate peers equals to that in a swarm, i.e.,

$$F(n,m,0,0) = m \times n, \qquad P(n,m,0,0) = 1. \quad (2)$$

When the swarm is under attack, the total number of peers is $n + k$. Each legitimate peer releases a valid connection to perform optimistic unchoking, so that in total n valid connections are released. The probability of obtaining a valid connection by a legitimate peer is close to $\frac{n}{n+k}$. The number of valid connections obtained by legitimate peers after the f rst round is

$$\begin{aligned} F(n,m,k,1) &= m \times n - n + \frac{n}{n+k} \times n \\ &= F(n,m,0,0) - n + \frac{n}{n+k} \times n, \quad (3) \end{aligned}$$

where $F(n,m,k,1)$ contains three parts: $F(n,m,0,0)$ is the number of valid connections in the f rst round, n is the number of valid connections released, and $\frac{n}{n+k} \times n$ is the number of valid connections relocated to legitimate peers. Next, we rewrite Eqn. (3) in a recursion form as

$$\begin{aligned} F(n,m,k,1) &= F(n,m,0,0) - n + \frac{n}{n+k} \times n \\ &= F(n,m,0,0) \times \frac{m-1}{m} + \frac{n}{n+k} \times n. \quad (4) \end{aligned}$$

The number of valid connections obtained by legitimate peers after the second round is

$$\begin{aligned} F\big(n,m,k,2\big) &= F\big(n,m,k,1\big) - \frac{F\big(n,m,k,1\big)}{m \times n} \times n \\ &\quad + \frac{n}{n+k} \times n \\ &= F\big(n,m,k,1\big) \times \frac{m-1}{m} + \frac{n}{n+k} \times n. \quad (5) \end{aligned}$$

Generally, the number of valid connections obtained by

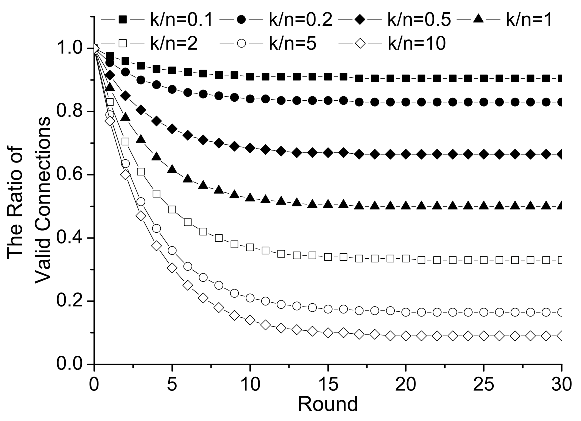

Fig. 3. The ratio of valid connections with different ratios of malicious peers vs. legitimate peers.

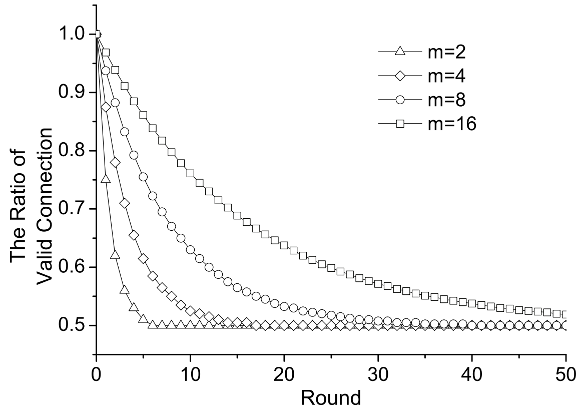

Fig. 4. The ratio of valid connections with different numbers of upload connections.

legitimate peers after the tth round is

$$\begin{aligned} F\big(n,m,k,t\big) &= F\big(n,m,k,t-1\big)\times\frac{m-1}{m}+\frac{n}{n+k}\times n \\ &= \frac{k\times n\times(m-1)^t+n^2\times m^t}{(n+k)\times m^{(t-1)}},\ t>0. \quad (6)\end{aligned}$$

Theorem 2. The ratio between the numbers of valid connections after and before an attack after the tth round optimistic unchoking is

$$P(n,m,k,t)=\frac{k}{n+k}\times(\frac{m-1}{m})^t+\frac{n}{n+k},\quad t>0. \quad (7)$$

Proof. The number of valid connections before an attack follows $F(n,m,0,0)=m\times n$, and the ratio between the numbers of valid connections after and before an attack is

$$\begin{aligned} P\big(n,m,k,t\big) &= \frac{F\big(n,m,k,t\big)}{F\big(n,m,0,0\big)}=\frac{k\times(m-1)^t+n\times m^t}{\big(n+k\big)\times m^t} \\ &= \frac{k}{n+k}\times\left(\frac{m-1}{m}\right)^t+\frac{n}{n+k},\ t>0. \quad (8)\end{aligned}$$

From Eqns. (1) and (7), we can see that $F(n,m,k,t)$ and $P(n,m,k,t)$ depend on several factors, which include the number of malicious peers k, the number of legitimate peers n, the number of upload connections of each legitimate peer m, and the number of round t. When m goes to inf nity, $P(n,m,k,t)$ approaches to one. This means that if there are a large number of upload connections, then the leechers attack has little impact. When the number of rounds t goes to inf nity, $P(n,m,k,t)$ approaches to $n/(n+k)$, which is the ratio between the number of legitimate peers and that of all peers (legitimate plus malicious).

Figs. 3 and 4 show the impacts of k/n and m on the ratio of valid connections. Fig. 3 shows the ratio for different values of k/n and a default number of upload connections ($m=4$). We can see that, the more malicious peers exist, the faster the ratio decreases. After 15 rounds, the curves become stable, showing that the ratio of valid connections is determined by the ratio of malicious peers and legitimate peers. Fig. 4 plots the ratio for different values of m when $k/n=1$, i.e., the number of malicious peers is the same as that of legitimate peers. Fig. 4 shows that, the more upload connections exist, the more valid connections are obtained by legitimate peers.

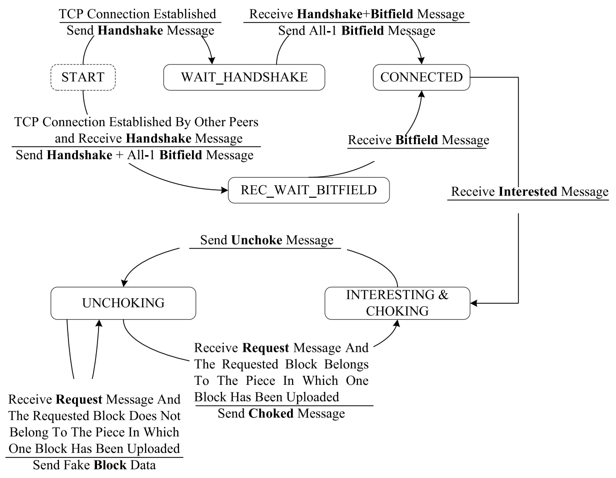
TCP Connection Established / Send **Handshake** Message
Receive **Handshake**+**Bitfield** Message / Send All-1 **Bitfield** Message
START
WAIT_HANDSHAKE
CONNECTED
TCP Connection Established By Other Peers and Receive **Handshake** Message / Send **Handshake** + All-1 **Bitfield** Message
Receive **Bitfield** Message
REC_WAIT_BITFIELD
Receive **Interested** Message
Send **Unchoke** Message
UNCHOKING
INTERESTING & CHOKING
Receive **Request** Message And The Requested Block Belongs To The Piece In Which One Block Has Been Uploaded / Send **Choked** Message
Receive **Request** Message And The Requested Block Does Not Belong To The Piece In Which One Block Has Been Uploaded / Send Fake **Block** Data

Fig. 5. The state diagram of fake-block attack.

B. Valid Bandwidth Model Based on Fake-Block Attack

In this subsection, we present a stochastic model for quantitative analyses of the impacts of fake-block attack on BT peers. The fake-block attack is illustrated in Fig. 5. Assume that each peer has the same bandwidth and computation capability, and peers arrive according to a Poisson process. The rate at which a peer leaves a swarm depends on its download speed. A peer does not abort until it obtains the entire f le, and it leaves after it f nishes the download. 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. Our stochastic model studies the attack effect in the equilibrium state.

Assume that there are N peers serving a given fil $\mathcal{F}$. $\mathcal{F}$ is divided into s pieces, or $\mathcal{F}=\mathcal{F}_1\cup\mathcal{F}_2\cup\cdots\cup\mathcal{F}_s, \mathcal{F}_i\cap\mathcal{F}_j=\emptyset$ for $i\neq j$, where $\mathcal{F}_i$ is the ith piece of $\mathcal{F}$. The size of a piece is p. One piece is further divided into c blocks, and each block is a transmission unit. There are n legitimate peers including seeds and leechers. New peers arrive according to a Poisson process with a rate of λ. The download bandwidth of each peer is b_d. The number of sources is suff cient to allow every peer to saturate download bandwidth. After a leecher has acquired all pieces of $\mathcal{F}$, it leaves immediately. After a swarm is stabilized, k malicious peers with a total upload bandwidth b_{attack}^{total} join and launch fake-block attack. The malicious peers

TABLE I
DEFINITIONS OF NOTATIONS USED IN THE ANALYSIS.

Notation	Def nition
n	Number of normal peers
k	Number of attacking peers
s	Number of pieces of f le $\mathcal{F}$
p	Size of a piece
c	Number of blocks in one piece
λ	Arrival rate of new peers
$\overline{n}$	Number of normal peers in equilibrium state
ω	Piece parallelism degree
b_d	Download bandwidth of a peer
b_{attack}^{total}	The total upload bandwidth of attacking peers
t_{normal}	Download duration without attack
t_{attack}	Download duration under attack
R_{delay}	Download delay ratio
$n_{complete}^{t}$	Instantaneous accomplishing peers at time t
b_d^{normal}	Received bandwidth of a normal peer from normal peers
b_d^{attack}	Received bandwidth of a normal peer from attacking peers
$b_d^{polluted}$	Consumed bandwidth of discarded content
μ	Polluted bandwidth diffusion coeff cient
p_{attack}	Probability to select an attacking source
p_{normal}	Probability to select a normal source

claim they have all s pieces of $\mathcal{F}$, and assume that they have suff cient bandwidth to accept requests from the other peers. The notations are given in Table I.

We introduce two indicators to evaluate attack effectiveness: (1) delay ratio is used to evaluate the pollution impact on a single peer; and (2) instantaneous accomplishing peers at a given time is used to evaluate the pollution impact on the entire swarm.

Definiti n 1 (Delay Ratio): Delay Ratio is defi ed by $R_{delay} = t_{attack}/t_{normal}$, where t_{normal} is the download time of a peer without fake-block attack, and t_{attack} is the download time of a peer in the same swarm when there is a fake-block attack. A larger value of R_{delay} indicates more serious attack impact is. If R_{delay} is very large, it means the peer can hardly fini h the download.

Definitio 2 (Instantaneous Accomplishing Peers): $n_{complete}^{t}$ is the number of instantaneous accomplishing peers at time t. When a swarm is in its equilibrium state, we have $n_{complete}^{t} = n^t \times [b_d^t/(sp)]$, where n^t is the number of legitimate peers at time t, and b_d^t denotes the average download bandwidth of all peers at time t. A smaller value of $n_{complete}^{t}$ means more serious impact on the swarm. This indicator ref ects the actual pollution degree of a swarm's bandwidth.

Before calculating the indicators, we need to defin two more parameters as follows.

Definitio 3 (Piece Parallelism Degree): If the blocks of one piece are downloaded from ω different sources, the Piece Parallelism Degree is ω.

Def nition 4 (Polluted Bandwidth Ratio): The bandwidth used by a legitimate user to download blocks from malicious peers is denoted as b_d^{attack}, and 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 defi ed as Polluted Bandwidth Ratio.

Theorem 3. The value of R_{delay} is given by

$$R_{delay} = \left(\frac{k+\overline{n}/2}{\overline{n}/2}\right)^{\omega}, \tag{9}$$

where attackers' bandwidth satisfie

$$\frac{b_{attack}^{total}}{n^t} \geq \left\{1 - \left(\frac{n^t/2}{k+n^t/2}\right)^{\omega}\right\} \times \frac{b_d}{\omega}. \tag{10}$$

Proof. In the text followed, we present some important equations of a stochastic model.

1) The download time of a peer without attack is given by
$$t_{normal} = s \times \frac{p}{b_d}, \tag{11}$$
where $s \times p$ denotes the size of f le $\mathcal{F}$.

2) The total number of peers in the equilibrium state is given in (12). The number of peers at time t is $dn(t) = \lambda \times dt - b_d/(sp) \times ndt$. We have $dn(t)/dt = 0$ when the swarm reaches its equilibrium state. Using (11), we have
$$\overline{n} = \lambda \times t_{normal}. \tag{12}$$

3) In the equilibrium state, the number of peers that possess a specifi piece is given by
$$\overline{X_1} = \overline{X_2} = \cdots = \overline{X_j} = \cdots = \overline{X_s} = \frac{\overline{n}}{2}. \tag{13}$$
In the equilibrium state, the numbers of peers at different accomplishing degrees are the same, and the total number of pieces is $\overline{n} \times s/2$. The piece selection strategy in BT makes all pieces evenly distributed, as shown in (13).

4) The bandwidth consumed by the pieces completely downloaded from malicious peers is given by (14). The probability of selecting a malicious peer is $k/(k+\overline{n}/2)$, and the probability to acquire a piece entirely from malicious peers is $p_{attack} = [k/(k+\overline{n}/2)]^{\omega}$. Hence, the total bandwidth consumed by those pieces is
$$b_d^{attack} = p_{attack} \times b_d = \left(\frac{k}{k+\overline{n}/2}\right)^{\omega} \times b_d. \tag{14}$$

5) The bandwidth consumed by the pieces completely downloaded from legitimate peers is given in (15). The probability of selecting a legitimate peer is $(\overline{n}/2)/(k+\overline{n}/2)$, and the probability to acquire a piece entirely from legitimate peers is $p_{normal} = [(\overline{n}/2)/(k+\overline{n}/2)]^{\omega}$. Hence, the total bandwidth consumed by those pieces is
$$b_d^{normal} = p_{normal} \times b_d = \left(\frac{\overline{n}/2}{k+\overline{n}/2}\right)^{\omega} \times b_d. \tag{15}$$

6) The polluted bandwidth ratio μ is given by
$$\mu = \frac{b_d^{polluted}}{b_d^{attack}} = \frac{b_d - b_d^{normal}}{b_d^{attack}} = \frac{b_d - (\frac{\overline{n}/2}{k+\overline{n}/2})^{\omega} \times b_d}{b_d^{attack}}. \tag{16}$$

7) The value of R_{delay} is given by
$$R_{delay} = \frac{t_{attack}}{t_{normal}} = \frac{\frac{s\times p}{b_d^{normal}}}{\frac{s\times p}{b_d}} = \left(\frac{k+\overline{n}/2}{\overline{n}/2}\right)^{\omega}. \tag{17}$$

8) The value of $n_{complete}^{t}$ is
$$n_{complete}^{t} = n^t \times \frac{b_d^{normal}}{sp} = \left(\frac{k+\overline{n}/2}{\overline{n}/2}\right)^{\omega} \times \frac{b_d}{sp} \times n^t. \tag{18}$$

Eqns. (17) and (18) indicate the attack effectiveness by fake-block attack. This requires that the bandwidth contributed by all attacking peers must fully satisfy content requests from the other peers. From (16), we can see that the bandwidth consumed by the pieces not completely downloaded from legitimate peers is $b_d^{polluted}$, and the attack bandwidth evenly allocated to each downloading peer is b^{total}_{attack}/n^t. The hash checking will fail even if one block is fake. Hence, the attack is most effective when there is one block of a piece coming from a malicious peer. The minimal required bandwidth is $b_d^{polluted}/\omega$. Therefore, attackers' bandwidth should satisfy

$$\frac{b^{total}_{attack}}{n^t} \geq \left\{1 - \left(\frac{n^t/2}{k + n^t/2}\right)^{\omega}\right\} \times \frac{b_d}{\omega}. \qquad (19)$$

If the bandwidth is not enough, attacking effectiveness will be lower than the results given in (17) and (18). Below, we list several factors that impact on pollution effectiveness.

1) Popularity of a f le: Eqn. (19) implies that the more the peers exist, 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 ref ects the popularity of a f le. Therefore, it is more diff cult to obtain the upper bound of pollution effectiveness for a popular f le.
2) The number of malicious peers: From (17), we can see that more malicious peers can generate a larger delay ratio, but at the same time may decrease the upload bandwidth of each malicious peer. It is important to select an appropriate value of k.
3) Piece parallelism degree: Eqn. (19) also shows that an attacker can obtain the required bandwidth more easily with a larger value of ω, which means a larger delay ratio.
4) 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.

Figs. 6 and 7 show the impacts of those factors on attack effectiveness. Fig. 6 plots Delay Ratio for different Piece Parallelism Degrees ω and different numbers of malicious peers. We set the f le size as 400 MB, arrive rate of new peers as one, and the download bandwidth of each peer as 400 KB/s. In the stable state, the total number of downloading peers is 1000. Fig. 6 shows that with a higher value of ω, it is easier to gain a larger Delay Ratio. A larger number of malicious peers also causes a higher Delay Ratio. However, when the upload bandwidth of each peer decreases to a value that can not meet the requirement of download requests, the Delay Ratio stops increasing. Fig. 7 plots the Delay Ratio for different swarm sizes (100 to 2000 peers) and different pollution bandwidths. In this case, we set the fil size to 400 MB, the download bandwidth of each peer to 400 KB/s, and the Piece Parallelism Degree ω to two. Fig. 7 shows that the Delay Ratio increases as pollution bandwidth becomes larger, and it increases very fast after upload bandwidth can satisfy the maximum download requests of all peers.

Our CP system was designed to control thousands of BitTorrent swarms. It is not practical to use unlimited machines and bandwidth at such a high cost, and thus an efficien method is necessary to determine how many resources should be properly allocated or deployed for each target swarm to maximize system efficien y. Based on above theoretical analysis, two proposed models are able to evaluate the cost and resource deployment of preventing pirate content sharing swarm. Our models can answer a general question: How many

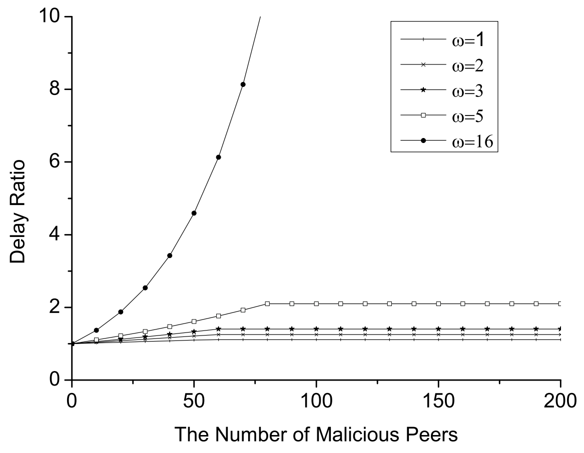

Fig. 6. Delay ratio with different numbers of malicious peers.

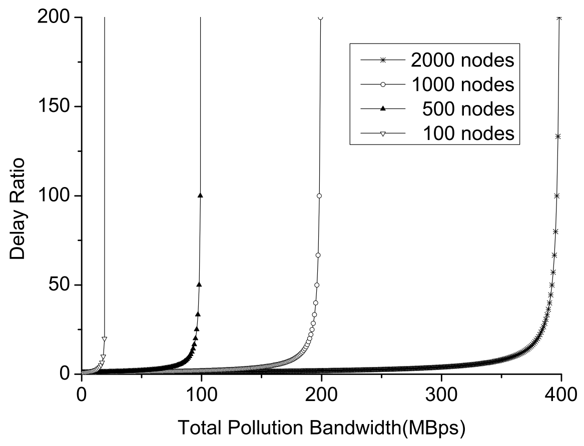

Fig. 7. Delay ratio with different pollution bandwidths.

resources (peers or bandwidth) should be used to delay the downloading duration of a 500-peers swarm more than three times longer than normal? Therefore, when a task is loaded into our CP system, it will measure the corresponding swarms to get necessary parameters, and then use these models to compute resource demand to guide resource allocation and deployment.

IV. Experimental Results

In addition to the theoretical evaluation given in Section III, we investigated the attack behaviors in the Internet using experiments.

A. Experimental Environment

We set up both public and semi-public environments to evaluate the system. The public environment consists of several servers with public IPs that can be connected directly from outside by two legitimate clients in the same network. Each server hosts thousands of lightweight clients behaving like leechers or fake-block peers. To compare the difference between attack and non-attack scenarios, we installed a f ltering (blacklist) function in one legitimate client. The filte ing function rejects attack connections. However, it is improper and illegal to directly control public torrents. In order to analyze attack impacts in a controlled environment, we also designed a semi-public environment that includes a private

TABLE II
COMPARISON OF DOWNLOAD TIMES.

Name	TV1		TV2 (BitComet)				MOVIE (BitSpirit)	
Size	191 MB		196.25 MB				519.25 MB	
Scenario	Non-attack	Attack	Non-attack	Attack			Non-attack	Attack
Ratio (Malicious/Legitimate)	0	≈ 2	0	≈ 2	≈ 7	≈ 11	0	≈ 2
T_D	0.6	1.65	1.93	15.17	11.6	41.56	2.02	$+\infty$

tracker that can be accessed locally by our own clients and several servers with internal IPs that cannot be connected from outside. As a result, attacking clients are limited and can only be connected by our clients.

The fake-block attack may cause a long time to download a f le. This makes the measurement inconvenient. Instead, we use a ratio to estimate download speed, or $T_D = (Duration/Download\ Percentage)$, where "Duration" is the download time recorded in the experiments, and "Download Percentage" is the percentage of the resource that has been downloaded. This equation can be used to estimate the remaining download task.

B. Real-World Results

Table II presents the experimental results obtained from a real network. The f rst column gives the download time in leechers attacks, in which malicious peers are almost two times as large as the number of legitimate peers. According to theoretical analysis, $P(n, m, k, t)$ will be $1/3$ in Eqn. (7) when t goes to inf nity, which means that the duration under attack is extended to three times longer. For actual attacks in the Internet, it is about 2.75 times ($1.65/0.6$). However, we also observed a wide f uctuation of download time (between one and f ve times). The possible reasons can be inferred as follows. Due to the dynamic nature of peers, the ratio between the numbers of malicious and legitimate peers is changing over time. The connection model does not consider different transmission rates of the connections.

The second and third columns present the contrast of download times between fake-block attack and non-attack scenarios in different clients. The results show that fake-block attack can extend to at least six times ($11.6/1.93$) of download time and cause serious performance degradation. The second column shows the effects depending on different ratios between malicious peers and legitimate peers, which means the more malicious peers exist, the longer the delay will be. The results also indicate that, compared with BitComet, BitSpirit is more vulnerable to fake-block attack. 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 malicious peers higher.

C. Impact Factors Analysis

Our study showed that several factors can influenc the effects of attack behaviors. Below, we discuss two important factors, i.e., swarm size and connection strategy.

Swarm size is def ned 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 attack because a peer can connect to many legitimate peers and get sufficie t 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. Fig. 8 shows the results of T_D using uTorrent in these two swarms. We can make the following observations. First, compared with the smaller swarm, the larger one has a shorter download time with a higher download rate, no matter whether the swarm is under attack or not. Second, our system can increase peer download duration at least three times for both small and large swarms. Third, the larger swarm is more diff cult to control. Fig. 9 demonstrates parallel download progress in legitimate and polluted uTorrent, and it shows an apparent delay in the polluted one. Also, we evaluated BitSpirit to see the difference between smaller and larger swarms. The polluted client can only obtain a small number of useful blocks in a long period of time, which means T_D is very large.

To demonstrate the impacts of connection strategies on the attacks, we choose three well-known clients: uTorrent, Vuze, and BitSpirit. First, these clients have huge populations with the coverage of more than 90% users. If our system succeeds in controlling their download processes, then it should be suitable for the majority of the population. Second, they are different in their design philosophies, especially for connection strategies that determine whether the connections should be dropped. BitSpirit maintains connections lazily for a long time until the other end drops them. Leechers attack can achieve a good delay ratio for BitSpirit with a low bandwidth usage. However, uTorrent uses a more progressive method to get a better download performance. uTorrent stops a connection if there are few activities or low traffic and this increases the diff culties to control it. uTorrent has naturally the immunity against leechers attacks. Under fake-block attacks, uTorrent demands more bandwidth and our clients try to keep pollution connections, which has a side effect to make our clients more likely being blacklisted.

We downloaded the larger swarm's torrent with three clients to compare their download efficie cies, and the results are given in Table III. Under attacks, uTorrent is the fastest client to fi ish the download among them with over four times longer than the duration in normal situation. In contrast, BitSpirit suffers seriously from infinit ve delay because attackers succeeded in occupying its download slots. Delay effect of Vuze is better than uTorrent, but worse than BitSpirit. This indicates that the progressive strategy indeed can resist malicious behaviors to a certain degree. They are depicted in Figs. 10, 11, and 12, respectively.

In addition, we show the bandwidth usage of our system to control different clients in Fig. 13. First, outbound traff c (around 100 KB/s) is higher than inbound traff c (around 60 KB/s), because the outbound traff c is responsible for

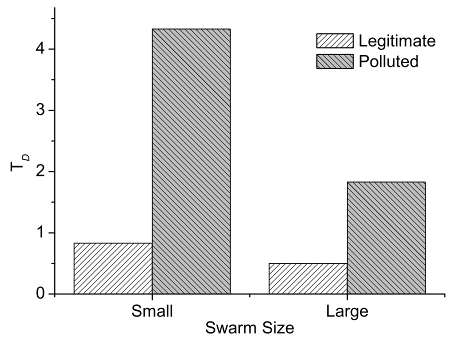

Fig. 8. T_D with different swarms using uTorrent.

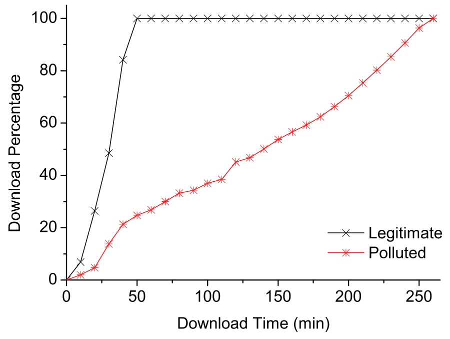

Fig. 9. The parallel download progress of uTorrent.

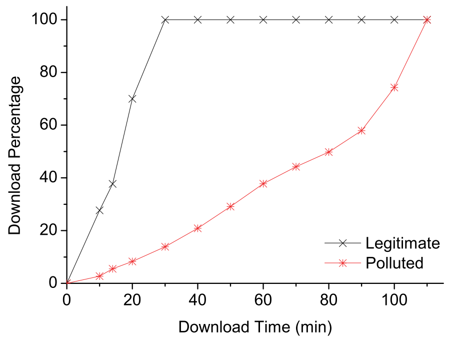

Fig. 10. The parallel download progress of uTorrent.

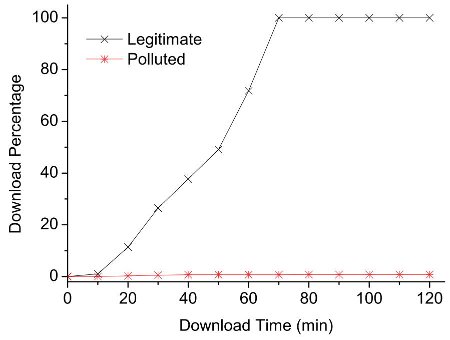

Fig. 11. The parallel download progress of Vuze.

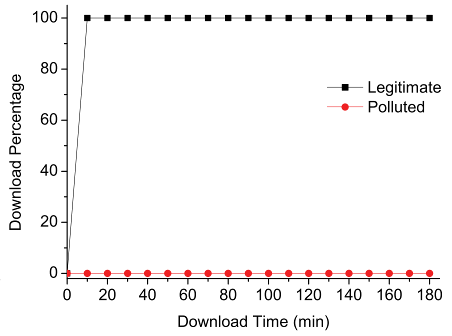

Fig. 12. The parallel download progress of BitSpirit.

TABLE III
COMPARISON OF DOWNLOAD TIME USING DIFFERENT CLIENTS.

Client	uTorrent	Vuze	BitSpirit
Legitimate (T_D)	0.83	1.67	0.17
Polluted (T_D)	4.33	250	$+\infty$

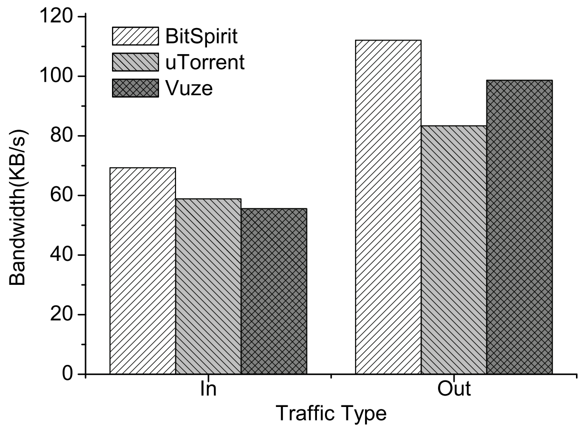

Fig. 13. Comparison of pollution traffic rates in different clients.

uploading fake blocks to victim clients. Second, BitSpirit and Vuze consume more bandwidth than uTorrent, implying that our system can not pollute many blocks in uTorrent. uTorrent can download the resource more quickly as expected.

V. CONCLUSION

In this paper, we conducted a study on copyright content protection in P2P networks. The study focused on the vulnerabilities of BT systems, such as leechers attacks and fake-block attacks, which can be used to delay the download progress of pirates. We carried out theoretical evaluations by analyzing several important parameters and modeling the attack behaviors. Compared to leechers attack, fake-block attack has a better delay ratio. In addition, we performed real-world experiments of attacking the existing torrents to evaluate the efficiencies in different clients. Our system succeeds in prolonging the download time to more than three times in most cases. Among those popular BT clients, uTorrent is the most difficult to control with many advanced features against the existing attacks. Our experimental results demonstrated that BitSpirit and Vuze suffer seriously from the attacks.

REFERENCES

[1] Envisional, “An estimate of infringing use of the Internet,” [Online]. Available: http://documents.envisional.com/docs/Envisional-Internet_Usage-Jan2011.pdf.
[2] BT@China Union. [Online]. Available: http://bt.btchina.net/.
[3] Mininova. [Online]. Available: http://www.mininova.org/.
[4] DMR 2011. [Online]. Available: http://www.ifpi.org/content/section_resources/dmr2011.html.
[5] Peer Media Technologies. [Online]. Available: http://peermediatech.com.
[6] Safenet. [Online]. Available: http://www.safenet-inc.com.
[7] E. Rescorla, “Introduction to distributed hash tables,” *IETF-65 Technical Plenary*, Mar. 2006.
[8] Peer Exchange. [Online]. Available: http://en.wikipedia.org/wiki/Peer_exchange.
[9] Cohen, “Incentives build robustness in BitTorrent,” in *Proc. 1st Workshop Economics Peer-to-Peer Syst.*, 2003.
[10] K. P. Chow, K. Y. Cheng, L. Y. Man, P. K. Y. Lai, L. C. K. Hui, C. F. Chong, K. H. Pun, W. W. Tsang, H. W. Chan and S. M. Yiu, “BTM - An automated rule-based BT monitoring system for piracy detection,” in *ICIMP '07*, 2007.
[11] J. Mee and P. A. Watters, “Detecting and tracing copyright infringements in P2P networks,” in *ICN/ICONS/MCL*, 2006.
[12] A. Sherman, A. Stavrou, J. Nieh, A. D. Keromytis, and C. Stein, “Adding trust to P2P distribution of paid content,” in *Proc. 12th International Conf. Inf. Security*, pp. 459-474, 2009.

[13] X. Li, S. Krishnan, and N. W. Ma, "A wavelet-PCA-based f ngerprinting scheme for peer-to-peer video f le sharing," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 3, pp. 365-373, 2010.

[14] D. Tsolis, S. Sioutas, A. Panaretos, I. Karydis, and K. Oikonomou, "Decentralized digital content exchange and copyright protection via P2P networks," in *ISCC 2011*, pp. 1056-1061, 2011.

[15] Q. Qiu, Z. Tang, and Y. Yu, "A decentralized authorization scheme for DRM in P2P f le-sharing systems," in *Consumer Commun. Netw. Conf.*, pp. 136-140, 2011.

[16] X. Zhang, D. Liu, S. Chen, Z. Zhang, and R. Sandhu, "Towards digital rights protection in bittorrent-like P2P systems," in *Proc. 15th ACM/SPIE Multimedia Comput. Netw.*, 2008.

[17] X. Lou and K. Hwang, "Collusive piracy prevention in P2P content delivery networks," *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 970-983, 2009.

[18] M. Yoshida, S. Ohzahata, A. Nakao, and K. Kawashima, "Controlling fil distribution in the share network through content poisoning," in *AINA '10*, pp. 1004-1011, 2010.

[19] C. Wang and C. Chiu, "Copyright protection in P2P networks by false pieces," in *Proc. 8th International Conf. Autonomic Trusted Comput.*, pp. 215-227, 2011.

[20] P. Dhungel, D. Wu, B. Schonhorst, and K. W. Ross, "A measurement study of attacks on BitTorrent leechers," in *P2P '08*, pp. 7-7, 2008.

[21] P. Dhungel, D. Wu, X. J. Hei, B. Schonhorst, and K. W. Ross, "Is BitTorrent unstoppable?" [Online]. Available: http://cis.poly.edu/~ross/papers/IsBTunstoppable.pdf.

[22] M. A. Konrath, M. P. Barcellos, and R. B. Mansilha, "Attacking a swarm with a band of liars: Evaluating the impact of attacks on BitTorrent," in *P2P '07*, pp. 37-44, 2007.

[23] BitSpirit. [Online]. Available: http://www.bitspirit.cc/.

[24] BitComet. [Online]. Available: http://www.bitcomet.com/index-zh-cn.php.

[25] uTorrent. [Online]. Available: http://www.utorrent.com/.

[26] Vuze. [Online]. Available: http://www.vuze.com/.

[27] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang, "Exploiting BitTorrent for fun (but not profit)" in *IPTPS '06*, 2006.

[28] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in BitTorrent networks with the large view exploit," in *IPTPS '07*, 2007.

[29] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Privacy-preserving P2P data sharing with OneSwarm," in *Proc. ACM SIGCOMM 2010 Conf.*, ACM, 2010, pp. 111-122.

Hongli Zhang received her BSc degree in Computer Science from Sichuan University, Chengdu, China in 1994, and her Ph.D. degree in Computer Science from Harbin Institute of Technology (HIT), Harbin, China in 1999. In 2012, she was a visiting scholar at North Carolina State University, Raleigh, NC, USA. She is currently a Professor in School of Computer Science and Technology in HIT and the Vice Director of National Computer Information Content Security Key Laboratory. Her research interests include network and information security, network measurement and modeling, and parallel processing.

Lin Ye received his BSc, MSc, and Ph.D. degrees in Computer Science from Harbin Institute of Technology (HIT), Harbin, China, from 2000 to 2011. In 2012, he was a visiting scholar at Temple University, Philadelphia, PA, USA. He is currently a Lecturer in School of Computer Science and Technology in HIT. His research interests include peer-to-peer measurement, cloud computing and information security.

Jiantao Shi received his BSc and MSc degrees in Computer Science from Harbin Institute of Technology (HIT), Harbin, China, from 1998 to 2004. He is currently a Ph.D. candidate in School of Computer Science and Technology in HIT. His research interests include network security and peer-to-peer networks.

Xiaojiang (James) Du is currently an Associate Professor in the Department of Computer and Information Sciences at Temple University. Dr. Du received his BE degree from Tsinghua University, China in 1996, and his MSc and Ph.D. degrees from the University of Maryland, College Park, in 2002 and 2003, respectively, all in Electrical Engineering. His research interests are security, systems, wireless networks and computer networks. Dr. Du has published over 100 journal and conference papers in the areas, and has been awarded more than $3M research grants from the US NSF and Army Research Off ce. He serves on the editorial boards of four international journals.

Hsiao-Hwa Chen is currently a Distinguished Professor in the Department of Engineering Science, National Cheng Kung University, Taiwan. He obtained his BSc and MSc degrees from Zhejiang University, China, and a Ph.D. degree from the University of Oulu, Finland, in 1982, 1985 and 1991, respectively. He is the founding Editor-in-Chief of Wiley's *Security and Communication Networks Journal* (http://www.interscience.wiley.com/security). He is also serving as the Editor-in-Chief for *IEEE Wireless Communications*. He is a Fellow of IEEE, IET, and BCS.