Attacks on BitTorrent – An Experimental Study

Marti Ksionsk[1], Ping Ji[1], and Weifeng Chen[2]

[1] Department of Math & Computer Science
John Jay College of Criminal Justice
City University of New York
New York, New York 10019
glassnickels@gmail.com,pji@jjay.cuny.edu

[2] Department of Math & Computer Science
California University of Pennsylvania
California, PA 15419
chen@calu.edu

Abstract. Peer-to-peer (P2P) networks and applications represent an efficient method of distributing various network contents across the Internet. Foremost among these networks is the BitTorrent protocol. While BitTorrent has become one of the most popular P2P applications, attacking BitTorrent applications recently began to arise. Although sources of the attacks may be different, their main goal is to slow down the distribution of files via BitTorrent networks. This paper provides an experimental study on peer attacks in the BitTorrent applications. Real BitTorrent network traffic was collected and analyzed, based on which, attacks were identified and classified. This study aims to better understand the current situation of attacks on BitTorrent applications and provide supports for developing possible approaches in the future to prevent such attacks.

1 Introduction

The demand for media content on the Internet has exploded in recent years. As a result, file sharing through peer-to-peer (P2P) networks has noticeably increased in kind. In a 2006 study conducted by CacheLogic [9], it was found that P2P accounted for approximately 60 percent of all Internet traffic in 2006, a dramatic growth from its approximately 15 percent contribution in 2000. Foremost among the P2P networks is the BitTorrent protocol. Unlike traditional file sharing P2P applications, a BitTorrent program downloads pieces of a file from many different hosts, combining them locally to construct the entire original file. This technique has proven to be extensively popular and effective in sharing large files over the web. In that same study [9], it was estimated that BitTorrent comprised around 35 percent of traffic by the end of 2006. Another study conducted in 2008 [4] similarly concluded that P2P traffic represented about 43.5 percent of all traffic, with BitTorrent and Gnutella contributing the bulk of the load.

During this vigorous shift from predominately web browsing to P2P traffic, concern over the sharing of copyrighted or pirated content has likewise escalated. The Recording Industry Association of America (RIAA), certain movie studios, and the Comcast ISP have attempted to block BitTorrent distribution of certain content or tracking BitTorrent users in hopes of prosecuting copyright violators. In order to curtail the exchange of pirated content through BitTorrent, opposing parties can employ two different attacks that can potentially slow the transfer of files substantially. The first is referred to as a fake-block attack, wherein a peer sends forged content to requesters. The second is an uncooperative peer attack, which consists of peers wasting the time of downloaders by continually sending keep alive messages, but never sending any content. These two attacks can also be used by disapproving individuals who just try to malfunction the BitTorrent system.

Not so many studies ([6,10]) have been conducted to understand the situation and consequences of such attacks. This paper aims to get a first hand look at the potential of fake-block and uncooperative-peer attacks, and to provide supports for developing possible approaches in the future to prevent such attacks. An experiment was set up to download files via BitTorrent applications, during which, BitTorrent traffic was captured and analyzed. We classified the hosts connected during the download process into different categories, and identified attack activities based on the traffic. We observed that the two different attacks mentioned above indeed exist within the BitTorrent. We also found that the majority of peers connected in downloading turn out to be completely useless for file acquisition. This process of culling through the network traces is useful in understanding the issues that cause delays in file acquisition in BitTorrent systems.

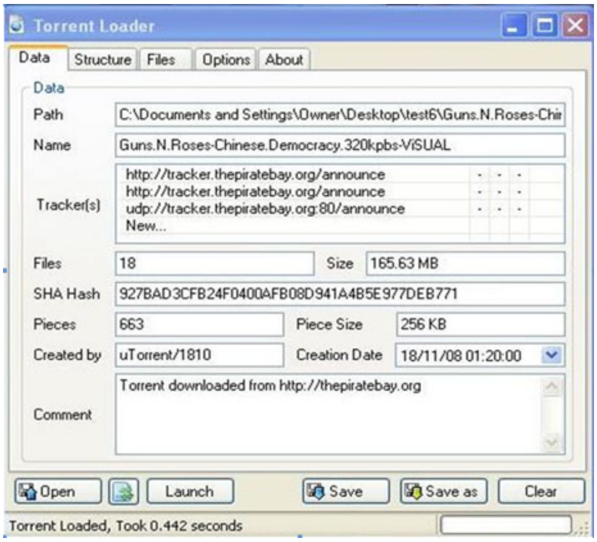
The rest of the paper is organized as follows. In Section 2, the BitTorrent protocol is explained and the two different attacks, fake-block attack and uncooperative peer attack, are thoroughly examined. Section 3 describes the experiment design and implementation. We present the experimental results and some discussion in Section 4. Finally, Section 5 concludes the paper.

2 BitTorrent Background and Attack Schemes

The BitTorrent protocol consists of four main phases. First, a torrent seed for a particular file is created and uploaded to search sites and message boards. Next, a person who is interested in the file downloads the seed and opens the seed using a BitTorrent client. Then, the BitTorrent client, based on the seed, contacts one or more trackers. Trackers serve as the first contact points of the client. They will point the client to other peers that already have all or some of the file requested. Finally, the client connects to these peers, receives blocks of the file from them, and constructs the entire original file. This section will describe these four stages in details, based on the BitTorrent protocol specification [5,8].

2.1 The Torrent Seed

The torrent seed provides a basic blueprint of the original file and specifies how the file can be downloaded. This seed is created by a user, referred to as the initial seeder, who has the complete data file. Typically, the original file is divided into 256kb pieces, though piece lengths between 64kb and 4mb are acceptable. The seed consists of an “announce” section, which specifies the IP address(es) of the tracker(s), and an “info” section, which contains file names, their lengths, the piece length used, and a SHA-1 hash code for each piece. The SHA-1 hash values for each piece included in the info section of the seed are used by clients to verify the integrity of the pieces they download. In practice, pieces are further broken down into blocks, which are the smallest units exchanged between peers. Figure 1 shows the information found in a torrent seed as displayed in a freely available viewer, TorrentLoader 1.5 [2].



Data

Path: C:\Documents and Settings\Owner\Desktop\test6\Guns.N.Roses-Chi

Name: Guns.N.Roses-Chinese.Democracy.320kpbs-ViSUAL

Tracker(s):
http://tracker.thepiratebay.org/announce
http://tracker.thepiratebay.org/announce
udp://tracker.thepiratebay.org:80/announce
New...

Files: 18 Size: 165.63 MB

SHA Hash: 927BAD3CFB24F0400AFB08D941A4B5E977DEB771

Pieces: 663 Piece Size: 256 KB

Created by: uTorrent/1810 Creation Date: 18/11/08 01:20:00

Comment: Torrent downloaded from http://thepiratebay.org

Torrent Loaded, Took 0.442 seconds

Fig. 1. Torrent File Information

After the seed is created, the initial seeder publishes it on torrent search engines or on message boards.

2.2 Acquiring Torrent Files

Before a user can search and download a file of interest, the user must first install one of several different BitTorrent (BT) clients that can process torrent seeds to connect to trackers, and ultimately other peers that have the file. A BitTorrent client is any program that can create, request, and transmit any type of data using the BitTorrent protocol. Clients vary slightly in appearance and implementation, but can be used to acquire files created by any other clients. Finding the torrent seeds is simply a matter of scanning known torrent hosting sites (such as thepiratebay, isohunt, or torrentz) or search engines. The user then downloads the seed and loads it into the client to begin downloading the file.

2.3 The Centralized Trackers

In BitTorrent systems centralized trackers serve as the first contact points for clients interested in downloading a particular file. IP addresses of the trackers' are listed in the torrent seed. Once a seed is opened in a BT client, the client will attempt to make connections with the trackers. The trackers will then verify the integrity of the seed and generate a list of peers that have a complete or partial copy of the file ready to share. This set of peers constitute the swarm of the seed. Every seed has its swarm. Peers in a swarm can either be seeders or leechers. Seeders are peers that are able to provide the complete file. Leechers are peers that do no yet have a complete copy of the file; however, they are still capable of sharing the pieces that they do have with the swarm. The tracker continually provides updated statistics about the number of seeders and leechers in the swarm.

The BitTorrent protocol also supports trackerless methods for file sharing, such as Distributed Hash Tables (DHT) or Peer Exchange methods. These decentralized methods are also supported by most BT clients. Under a decentralized method, the work of a traditional centralized tracker is distributed across all of the peers in the swarm. Decentralized methods increase the number of discovered peers. A user can configure his/her BT client to support centralized methods, or decentralized methods, or both. In this paper, we focuses solely on the centralized tracker model.

2.4 Joining the Swarm

In order for a new peer to join the swarm of a particular seed, the peer must attempt to establish TCP connections with other peers already in the swarm. After the TCP handshake, two peers then exchange a BitTorrent handshake. The initiating peer sends a handshake message containing a peer id, the type of the BT client being used, and an info_hash of the torrent seed. If the receiving peer responds with corresponding information, the BitTorrent session is considered open. Immediately after the BitTrorrent handshake messages are exchanged, each peer sends the other information about which pieces of the file it possesses. This exchange takes the form of bit-field messages with a stream of bits whose bit index corresponds to a piece index. The exchange is performed only once during the session. After the bit-field messages have been swapped, data blocks can begin to be exchanged over TCP. Figure 2 illustrates the BitTorrent handshake, while Figure 3 summarizes the exchange of data pieces between peers.

2.5 Peer Attacks on the Swarm

From the above description of the BitTorrent protocol, it is evident that someone can manipulate to delay the transmission of a file to an interested peer. The first attack, referred to as the Fake-Block Attack [6], takes advantage of the fact that a piece of a file is not verified via hash until it has been downloaded. Thus, attacking peers can send bad blocks of the file to interested parties, and

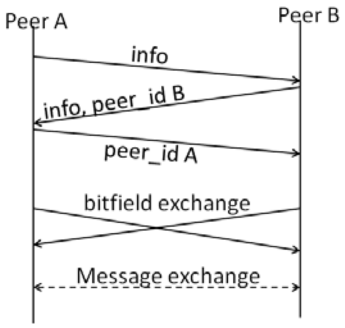


Fig. 2. The BitTorrent Handshake [7]

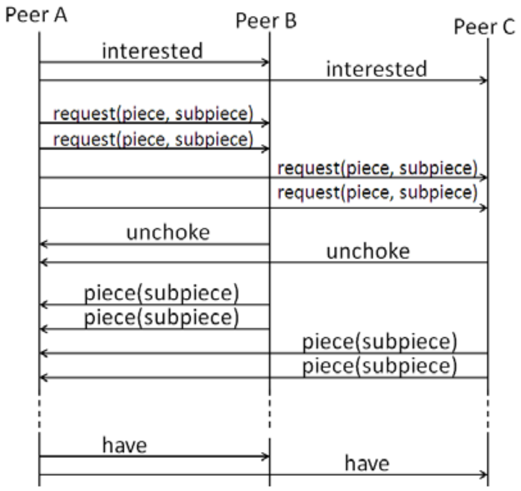


Fig. 3. BitTorrent Protocol Exchange [7]

when these blocks are combined with those from other sources, the completed piece will not be a valid copy since the piece hash will not match that of the original file. This piece will then be discarded by the client and will need to be downloaded again. While this generally only serves to increase the total time of the file transfer, swarms that contain large numbers of fake-blocking peers could potentially cause enough interference that some downloaders would give up.

The second attack is referred to as the Uncooperative, or Chatty, Peer Attack [6]. In this scheme, attacking peers exploit the BitTorrent message exchange protocol to hinder a downloading client. Depending on the client used, these peers can simply keep sending BitTorrent handshake messages without ever sending any content (as is the case in the Azereus client), or they can continually send keep-alive messages without delivering any blocks. Since the number of peer connections is limited, which is often set to 50, connecting to numerous chatty peers can drastically increase the download time of the content.

3 Experiment Design and Implementation

In this section, we describe the design and implementation of our experimental study. The design of this experiment is based heavily on the work in [6]. Three of the most popular album seeds (Beyonce_IAmSasha, GunsNRoses_Chinese, and Pink_Funhouse) were downloaded from thepiratebay.org for the purposes of this experiment. In order to observe the behavior of peers within the swarm and to identify any peers that might be considered attackers as defined in the two attack schemes previously, network traffic during the download process was captured. The traces were then analyzed, with data reviewed on a per host basis.

It is clear from the design of BitTorrent protocol that the efficiency of file distribution relies heavily upon the behavior of peers within the swarm. Peers that behave badly, either intentionally or unintentionally, can cause sluggish download times, as well as poisoned content in the swarm. For the purposes of this experiment, peers were categorized similarly to [6]. Hosts were sorted into different groups as follows:

Table 1. Torrent Properties

Torrent#	File Name	File Size	# of Pieces	Swarm Statistics	Protocol Used
1	Beyonce_IAmSasha	239mb	960	1602	Centralized Tracker
2	GunsNRoses_Chinese	165.63mb	663	493	Centralized Tracker
3	Pink_Funhouse	186.33mb	746	769	Centralized Tracker

– No-TCP-connection Peers: peers with which a TCP connection cannot be established.
– No-BT-handshake Peers: peers with which a TCP connection can be established, but with which a BitTorrent handshake cannot be established.
– Chatty Peers: peers that merely chat with our client. In this experiment, these peers establish a BitTorrent handshake and then only send out BitTorrent continuation data, not any data blocks.
– Fake-Block-Attack Peers: peers that upload forged blocks. These peers are identified by searching hash fails by pieces after the session is completed and then checking which peers uploaded fake blocks for particular pieces.
– Benevolent Peers: peers that communicate normally and upload at least one good block.
– Other Peers: peers that do not fit any of the above categories. This included clients that disconnected during the BT session before sending any data blocks and clients that never sent any data but did receive blocks from the test client

The experiment was implemented using an AMD 2.2 GHz machine with 1GB of RAM, connected to the Internet via a 100 Mbps DSL connection. The three seeds were loaded into the BitTorrent v.6.1.1 client. Based on the seeds, the client connected to trackers and the swarm. Within the client, only the centralized tracker protocol was enabled; DHT and Peer Exchange were both disabled. During each of the three download sessions for the three albums, Wireshark [3] was used to capture network traces, and the BT client's logger was also enabled to capture data for hash fails during a session. A network forensic tool, NetworkMiner [1], was then used to parse the Wireshark data to determine the number of hosts, as well as their IP addresses. Finally, traffic to and from each peer listed in NetworkMiner was examined using filters within Wireshark to determine which category listed above the traffic belonged to.

The properties of the three torrent seeds used in this experiment are shown in Table 1. All three of the torrent seeds listed the same three trackers; however, during the session, only one of the tracker URLs was valid and working. The swarm statistics published in the seed are based on that single tracker.

4 Experiment Results

In this section, we present the experimental results and discuss our observations.

4.1 Results

The three albums were all downloaded successfully, though all three did contain hash fails during the downloading process. Chatty peers were also present in all three swarms. The results of each download are illustrated in Table 2.

Table 2. Download Results

Torrent #	Total Download Time	# Peers Contacted	Hash Fails
1	1 hour 53 minutes	313	21
2	33 minutes	203	2
3	39 minutes	207	7

The classifications of the peers found in the swarm varied only minimally from one seed to another. No-TCP-Connection peers accounted for by far the largest portion of the total number of peers in the swarm. There were three different observable varieties of No-TCP-Connection peers: the peer that never responded to the SYN sent from the initiating client, the peer that sent a TCP RST in response to the SYN, and the peer that sent an ICMP destination unreachable response. Of these three categories, peers that never responded to the initiator's SYN accounted for the bulk of the total. While sending out countless SYN packets without ever receiving a response or receiving only a RST in return certainly utilizes bandwidth that could be otherwise used to establish sessions with active peers, it is important to note that these No-TCP-Connection peers are not necessarily attackers. These peers included NATed peers, firewalled peers, stale IPS returned by trackers, and peers that have reached their TCP connection limit (generally set around 50) [6].

No-BT-Handshake peers similarly fell into two distinct groups: peers that completed the TCP handshake but did not respond to the initiating client's BitTorrent handshake, and peers with whom the TCP connection was ended by the initiating client (via TCP RST) prior to the BitTorrent handshake. The latter case is likely due to a limit on the number of simultaneous BitTorrent sessions allowed per peer. Furthermore, the number of times that the initiating client would re-establish the TCP connection without ever completing a BT handshake ranged from 1 to 25. Clearly, the traffic generated while continually re-establishing TCP connections uses up valuable bandwidth that could be utilized by productive peers.

In this experiment, Chatty peers were classified as such when they repeatedly sent BitTorrent continuation data (keep-alive packets) without ever sending any data blocks to the initiating client. Generally in these connections, the initiator would continually send HAVE piece messages to the peer and would receive only TCP ACK messages in reply. Also, when the initiator would request a piece that the peer revealed that it owned in its initial bitfield message, no response would be sent. In this case, a Chatty peer kept open unproductive BitTorrent sessions that could otherwise have been used for other cooperative peers.

Table 3. Peer Classifications

	No-TCP-Connection			No-BT-Handshake					
Torrent #	No SYN ACK	RST	ICMP	No Handshake Response	RST	Fake Block	Chatty	Benevolent	Other
1	136	43	9	15	19	11	16	57	4
2	90	23	5	13	28	1	4	39	1
3	106	18	6	15	23	2	5	32	0
Total	332	84	20	43	70	14	25	128	5

The number of fake blocks discovered in each swarm varied quite widely, as did the number of unique peers who sent the false blocks. The first seed had 21 different block hash fails that were sent from only 11 unique peers. Among these 21 failed blocks, 9 of them came from a single peer. The other two seeds had far fewer hash fails, but the third seed showed a similar pattern – of the 7 hash fails, 6 were sent by the same individual peer.

The complete overview of peer classification for each torrent is exhibited in Table 3. From this table, it is evident that in all cases the majority of contacted peers in the swarm were not useful to the initiating client. Whether the peer actively fed fake content into the swarm, or merely inundated the client with hundreds of useless packets, all were responsible for slowing the exchange of data throughout the swarm. Figures 4 and 5 show the distribution of each type of peers in the swarms of each seed, as well as the combined distribution across all of the three seeds.

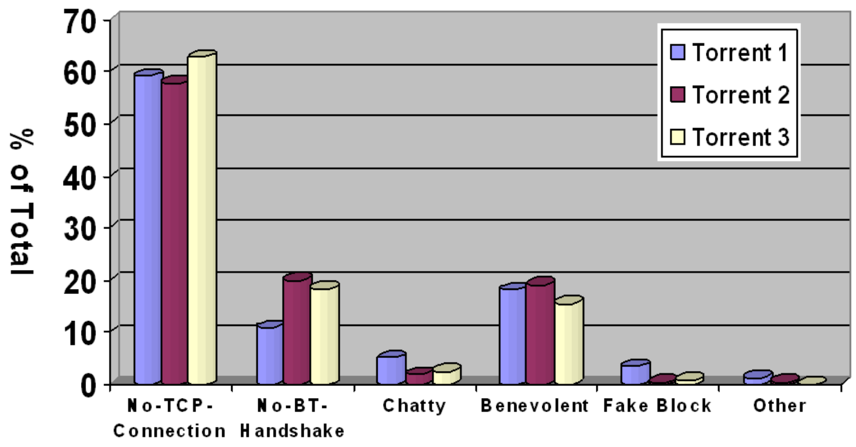


Fig. 4. Peer Classifications by Percent of Total

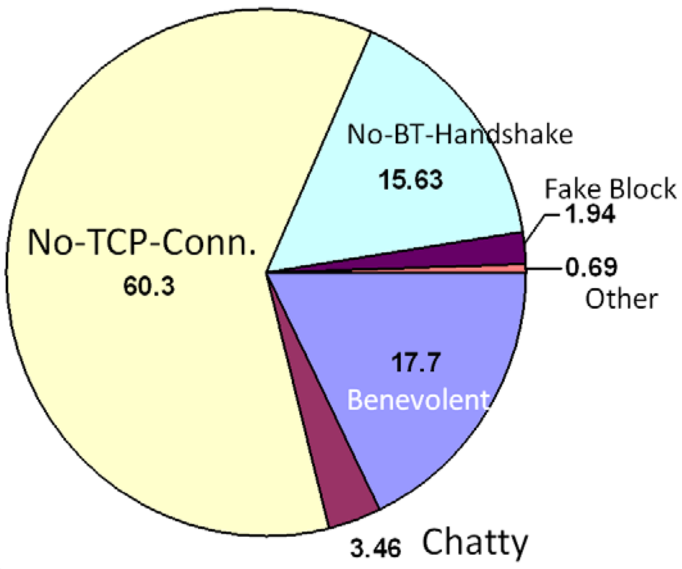


Fig. 5. Peer Distribution Combined Over all Torrents

4.2 Discussion

The experiment yielded interesting results. First, the analysis of network traces during a BitTorrent session demonstrated that while uncooperative/chatty peers do exist within the swarm, they are present in fewer numbers than anticipated. This may be due to the BitTorrent client used, as flaws in the Azereus client allow multiple BT Handshake and bitfield messages to be sent, whereas the client we used does not. The chatty peers observed in this experiment merely sustained the BT session without ever sending any data blocks. While these useless sessions definitely used up a number of the allocated BT sessions, the impact was mitigated by the small quantity of chatty peers relative to the total number of peers in the swarm. However, it can be concluded from these results that if a larger number of chatty peers reside in a single swarm, they can drastically slow download times of a file, since the BitTorrent client does not have a mechanism to detect and end sessions with chatty peers.

From this experiment it can also be seen that Fake-Block attackers indeed exist within the swarms of popular files. The first and third seeds provided perfect examples of the amount of time consumption a single attacking peer can have in a swarm. In both of these cases, one individual peer provided numerous fake blocks to the client. In the first seed, a single peer uploaded 9 failed blocks whereas in the third seed, another single peer uploaded 6 failed blocks. This caused the client to obtain those blocks from other sources after the hash check of the entire piece failed. After the attacking peer in the first seed had sent more than one fake blocks, the connection should have been disconnected to prevent any more time and bandwidth drain. However, the client has no mechanism to recognize which peers have uploaded fake blocks, and should therefore be disconnected. In a swarm with a small number of peers (e.g., a less popular file), a Fake-Block attacker could slow the transfer considerably as more blocks would need to be downloaded from the attacker. There do exist lists of IP addresses associated with uploading bad blocks that can be used to filter traffic in the BT client, but it is difficult to keep those lists updated as the attackers continually change addresses to avoid being detected.

Finally, the results of this experiment illustrated that the majority of peers that were contacted in the swarm turned out to be completely useless for the download. The number of No-TCP-Connection and No-BT-Handshake peers identified during each download was dramatic. While this is not in and of itself surprising, the number of times that the BT client tried to connect to a non-responding peer, or re-establish a TCP connection with a peer that never returns a BT handshake is striking. In some cases, 25 TCP sessions were opened even though the BT handshake was never once returned. TCP SYN messages were sent continually to peers that never once responded or only sent RST responses. In very large swarms such as those in this experiment, it is not necessary to keep attempting to connect with non-responsive peers since there are so many others that are responsive and cooperative.

5 Conclusions

In this paper, we have conducted an experimental study to investigate attacks on BitTorrent applications, which has not yet attracted much research attention. We have designed and implemented the experiment. BitTorrent traffic data has been captured and analyzed. We identified both fake-block attack and uncooperative/chatty attack based on the traffic. We also found that the majority of peers connected in downloading turned out to be completely useless for file acquisition. This experiment would help us to better understand the issues that cause delays in file download in BitTorrent systems. By identifying peer behavior that is detrimental to the swarm, this study is an important exercise to contemplate modification to BitTorrent clients and to develop possible approaches in the future to prevent such attacks.

Acknowledgments. This work is supported in part by National Science Foundation grant CNS-0904901 and National Science Foundation grant DUE-0830840.

References

1. NetworkMiner, `http://sourceforge.net/projects/networkminer/`
2. TorrentLoader 1.5 (October 2007), `http://sourceforge.net/projects/torrentloader/`
3. WireShark, `http://www.wireshark.org/`
4. Sandvine, Incorporated. 2008 Analysis of Traffic Demographics in North American Broadband Networks (June 2008), `http://sandvine.com/general/documents/Traffic_Demographics_NA_Broadband_Networks.pdf`
5. Cohen, B.: The BitTorrent Protocol Specification (February 2008), `http://www.bittorrent.org/beps/bep_0003.html`
6. Dhungel, P., Wu, D., Schonhorst, B., Ross, K.: A Measurement Study of Attacks on BitTorrent Leechers. In: The 7th International Workshop on Peer-to-Peer Systems (IPTPS) (February 2008)
7. Erman, D., Ilie, D., Popescu, A.: BitTorrent Session Characteristics and Models. In: Proceedings of HET-NETs 3rd International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks, West Yorkshire, U.K (July 2005)
8. Konrath, M.A., Barcellos, M.P., Mansilha, R.B.: Attacking a Swarm with a Band of Liars: Evaluating the Impact of Attacks on BitTorrent. In: Proceedings of IEEE P2P, Galway, Ireland (September 2007)
9. ParkerK, A.: P2P Media Summit. CacheLogic Research presentation at the First Annual P2P Media Summit LA, `dcia.info/P2PMSLA/CacheLogic.ppt` (October 2006)
10. Pouwelse, J., Garbacki, P., Epema, D.H.J., Sips, H.J.: The bittorrent P2P file-sharing system: Measurements and analysis. In: van Renesse, R. (ed.) IPTPS 2005. LNCS, vol. 3640, pp. 205–216. Springer, Heidelberg (2005)