POSTER: Destabilizing BitTorrent's Clusters to Attack High Bandwidth Leechers

Florian Adamsky
City University London
London, England
florian.adamsky.1@city.ac.uk

Hassan Khan
SEECS, NUST
Islamabad, Pakistan
hassan.khan@seecs.nust.edu.pk

Muttukrishnan Rajarajan
City University London
London, England
r.muttukrishnan@city.ac.uk

Syed Ali Khayam
SEECS, NUST
Islamabad, Pakistan
ali.khayam@seecs.nust.edu.pk

Rudolf Jäger
THM University of
Applied Sciences
Campus Friedberg, Germany
rudolf.jaeger@iem.thm.de

ABSTRACT

BitTorrent protocol incentivizes sharing through its choking algorithm. BitTorrent choking algorithm creates clusters of leechers with similar upload capacity to achieve higher overall transfer rates. We show that a malicious peer can exploit BitTorrent's choking algorithm to reduce the upload utilization of high bandwidth leechers. We use a testbed comprising of 24 nodes to provide experimental evidence of a distributed attack in which the malicious peers increase the download time for high bandwidth leechers by up to 16 % and increases average download time of the swarm by up to 15 % by using distributed and loosely-coupled malicious peers which comprise only 4.7 % of the swarm. The countermeasures of this attack are a part of our ongoing research work.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Security

Keywords

Attack, BitTorrent, Choke Algorithm, Peer-to-Peer

1. INTRODUCTION

BitTorrent is one of the most popular P2P protocols and it comprises the largest share of P2P traffic on the Internet [5]. Consequently, BitTorrent is a target of many attacks launched by anti-P2P companies which are working closely with the music, television and film industries [4]. Active measurement based studies on live torrents on Internet prove the existence of such attacks [2].

In this paper we show that a malicious peer can exploit the incentive-based sharing mechanism of BitTorrent to attack high bandwidth leechers. The incentive-based sharing mechanism of BitTorrent is based on the choking algorithm which creates clusters of leechers with similar upload capacity and results in higher overall data transfer rates among leechers with higher bandwidth. A malicious peer can launch an attack to destabilize the clusters in a swarm to decrease intra-cluster communication, which results in a decrease in data transfer among high bandwidth leechers and increases the average download time of the swarm.

We provide empirical evidence of the efficacy of this attack using a testbed of 24 nodes. In our experimental evaluations, we first demonstrate that the malicious peer can successfully destabilize clusters of leechers with similar bandwidth. This destabilization of clusters results in up to 6 % decrease in intra-cluster communication. We then show that a set of distributed and loosely-coupled malicious nodes which comprises only 4.7 % of the swarm results in an up to 16 % increase in download time for high bandwidth leechers and an up to 15 % increase in average download time of the swarm. The aim of this paper is to expose a weakness in BitTorrent protocol's choking algorithm so that the scientific community can find countermeasures to this attack.

2. BACKGROUND

In this section we first introduce the BitTorrent terminology used in this paper. We then briefly discuss BitTorrent's choking algorithm.

Swarm: All the peers sharing a torrent are called a swarm.

Leecher: A peer is a leecher when it is downloading content of a torrent.

Seeder: A peer is a seeder when it has downloaded all the content and is sharing the content with other leechers.

Peer Set: Each peer maintains a list of other peers that it knows about within a swarm. This list is known as peer set.

Active Peer Set: Active peer set for a peer is the subset of its peer set that it can send data to.

Interested and Choked: A peer P is interested in peer Q when peer Q has some contents that peer P does not have. Peer P is choked by peer Q when peer Q does not want to send data to peer P.

With the necessary terminology chalked out, we now discuss the choking algorithm. A leecher determines its active peer set through the choking algorithm. The choking algorithm works in rounds of 10 seconds and during its execution in each round, the choking algorithm works in the following way [1]:

1. The leecher orders the peers in its peer set according to their upload rate and ignores peers who have not sent any data to it.

2. The leecher then unchokes $n-1$ peers, where n is the number of parallel uploads the leecher is allowed.

3. After every three executions of choking algorithm, a peer is chosen at random and optimistically unchoked.

The choking algorithm works in a tit-for-tat-ish way and favors the leechers who upload. Consequently, leechers will more frequently unchoke other leechers with similar upload capacities. This reciprocation of data rates results in convergence towards good clustering by grouping of leechers with similar upload capacity. Good clustering ensures that the leechers with high upload bandwidth are rewarded with higher download rates and they are able to complete their downloads more quickly. Finally, the choking algorithm has been shown to converge towards good clustering shortly after the beginning of the download [3]. We refer interested readers to [1] for a detailed description of the choking algorithm.

3. EXPERIMENTAL SETUP

We perform experiments using private torrents in our testbed consisting of 24 nodes and a controller and monitor node. The nodes are desktop machines which are running BitTorrent client Transmission[1] v. 2.3 over CentOS 5.3. We selected Transmission client since it provides a powerful command-line interface (CLI) which can be used to throttle traffic using shell scripts. In order to simultaneously control all the nodes, we make use of the parallel-ssh [2] client. We wrote several scripts to monitor and record the status of each peer at every second. This information includes nodes in peer set, nodes in active peer set, interested and choked states for each of the peers in the active peer set, and upload and download rate with each of the peers in the active peer set. The controller executes the experiments and monitors each of these nodes.

In our experiments, we divide the peers into three different classes based on their upload and download rate limits. The fast class has peers with 2000 kbps bandwidth (class 1), the medium class has peers with 1500 kbps bandwidth (class 2), and the slow class has peers with 1000 kbps bandwidth (class 3). It should be noted that these bandwidth limits are for both the upload rate and the download rate. During our experiments, each peer class had the same number of peers and the upload and download bandwidth within each of these peer classes were not changed. A file of 887 MiB was seeded by a seeder and all the peers joined the torrent at the same time. This created a flash crowd scenario which resulted in reproducible results. To create a more realistic scenario, on every node some random HTTP background traffic was also generated. Finally, every leecher disconnected after receiving a complete copy of the file and only the initial seeder stayed connected for the complete duration of the experiment. Finally, every experiment was repeated 10 times and the average values for these experiments are reported in subsequent sections.

[1]`http://www.transmissionbt.com/`
[2]`http://code.google.com/p/parallel-ssh`

4. ATTACK DETAILS AND EXPERIMENTAL EVALUATION

We now show how a malicious peer can exploit the clustering algorithm to increase the download time for high bandwidth leechers. Legout et al. [3] showed that the clustering produced by the choking algorithm favors leechers with higher upload bandwidth. The malicious leecher [3] perturbs the clustering produced by the choking algorithm. To this end, the malicious leecher L_{ar} provides high upload bandwidth whenever it is optimistically unchoked by the attackee leecher L_{ae}. Furthermore, L_{ar} keeps on providing L_{ae} with high upload bandwidth for a few rounds. Consequently, L_{ae} adds L_{ar} to its active peer set and changes the state of one of the existing leechers L_{ex} from unchoked to choked. After providing high upload bandwidth for a few rounds to L_{ae}, L_{ar} drops its upload bandwidth significantly. Meanwhile, L_{ex} has found another leecher which it has unchoked and added to its active peer set. Now it would be at least three more rounds before L_{ae} and L_{ex} could transfer data among themselves (using optimistic unchoke). Since data transfer between L_{ae} and L_{ex} would have resulted in the most optimum data exchange between the two, the malicious leecher managed to trick them into forming a cluster with a peer which *probably* did not belong to the same cluster as L_{ae} or L_{ex}. Continued attacks by L_{ar} on the high bandwidth leechers destabilizes the clusters and results in longer download times for the higher bandwidth leechers.

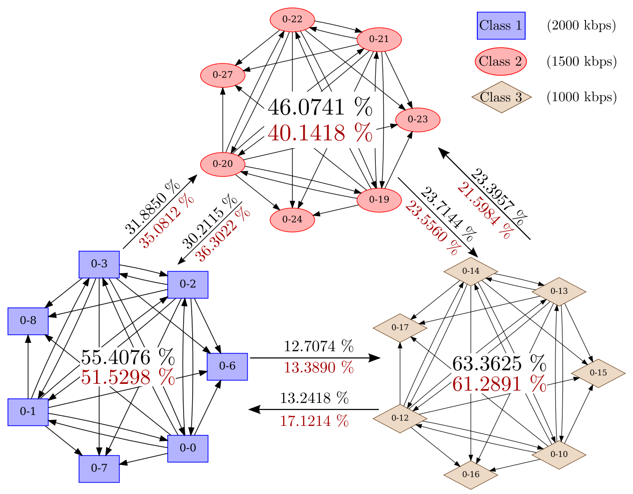

Figure 1: Affect of attack on intra- and inter-cluster activity. Arrows show the direction of data transfer. Seeder is omitted for simplicity (results without malicious peer are given in black; results with 1 malicious peer are given in red)

[3]We use the term malicious peer and malicious leecher interchangeably

We first demonstrate that the malicious peer destabilizes the high bandwidth cluster. To this end, we first allow all the peers to download the file without any malicious peer and note the intra- and inter-cluster activity (data transfer). We then introduce a malicious peer and repeat the same experiment. The malicious peer behaves like a high bandwidth leecher for 35 seconds with an upload bandwidth of 2000 kbps and after 35 seconds (one optimistic unchoke cycle + grace period) malicious peer drops its bandwidth to 20 kbps. To compare the results of these experiments, we first plot the percentage of intra- and inter-cluster activity without the malicious peer in Figure 1 (the values without malicious peer are displayed in black color). Due to space constraints, we plot the results with the malicious peer in the same figure using red color. Figure 1 shows that without the malicious peer, the intra-cluster activity is much more than the inter-cluster activity. However, after introducing the malicious peer, the intra-cluster activity drops by up to 6 % and the inter-cluster activity increases by up to 6 %.

We now determine the affect of destabilization of clusters on different classes of leechers with respect to file download time. To this end, we first download the file without any malicious peer and compare the download time for different classes of leechers when malicious peers are introduced. It can be observed from Figure 2 that when malicious peers are introduced, the download time for the fast leechers increases by up to 16 %. This is because the fast leechers are no longer transferring data with peers of similar (higher) bandwidth.

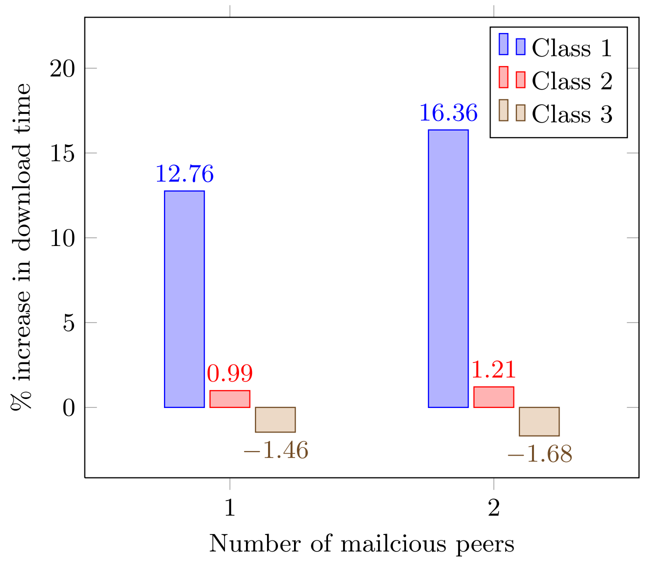

Figure 2: Increase/decrease in download time for different peer classes when malicious peers are introduced

The affect on medium leechers is a mere increase of 1 % in download time. This happens because the malicious peer forces the medium leechers to not only communicate with the fast leechers but also to communicate with the slow leechers. Therefore, for the medium leechers the communication with slow leechers almost neutralizes their data exchange with the fast leechers. Finally, the slow leechers are able to download the file in 1.5 % less time. This is because the slow leechers are now communicating more with the leechers of higher bandwidth.

5. FUTURE WORK

In this paper we demonstrated a malicious peer which attacks on a swarm and results in increased download time for fast leechers. While countermeasures is a part of our ongoing research work, a simple yet less elegant solution would be to keep a blacklist of IP addresses which are mounting such attack. Such blacklisted peers should always be avoided during random selection of the choking algorithm. As additional future work, we are planning to test this attack in PlanetLab. We also plan to investigate scalability of this attack when more malicious peers are added to the swarm.

6. ACKNOWLEDGEMENT

This work has been supported in parts by Pakistan Higher Education Commission. We would like to thank the National University of Science and Technology for providing us with the cluster we've used in our work.

7. REFERENCES

[1] B. Cohen. The Bittorrent Protocol Specification, Feb. 2008. `http://bittorrent.org/beps/bep_0003.html`.

[2] P. Dhungel, D. Wu, X. Hei, B. Schonhorst, and K. W. Ross. A Measurement Study of Attacks on BitTorrent Leechers. *IPTPS*, 2008.

[3] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in BitTorrent systems. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 301–312, New York, USA, 2007. ACM.

[4] Media Defender, March 2011. `http://www.mediadefender.com`.

[5] E. Van Der Sar. BitTorrent Traffic Surges After LimeWire Shutdown. `http://goo.gl/UIRmS`, March 2011.