Evaluating the Effectiveness of a BitTorrent-driven DDoS Attack

Jurand Nogiec
University of Illinois

Fausto Paredes
University of Illinois

Joana Trindade
University of Illinois

1. Introduction

BitTorrent is a widespread peer-to-peer file-sharing protocol that is becoming more known among users over the Internet. This popularity that has been surrounding it makes it vulnerable to organize and deploy a Distributed-Denial-of-Service attack on hosts over the Internet by exploiting vulnerabilities present in the BitTorrent protocol [2, 3].

The main objective of this project is to evaluate the effectiveness of a BitTorrent-driven DDoS Attack. The attack in [2], proposes a modification on BitTorrent trackers to enable this Denial-of-Service. Basically, peers in a BitTorrent swarm trust the tracker to get the list of peers that they could connect to, to download the file expected. The attack changes the implementation of the tracker to tell all peers that an arbitrary host is a peer in the swarm, leading to a DDoS attack. For the evaluations of the attack we use the network simulator ns-2.

2. Background

BitTorrent was developed by Bram Cohen and released in 2001 [4]. Its architecture is essentially composed of peers (*i.e.,* client applications that download and distribute files), and several central servers that coordinate connections between peers in a "swarm" (*i.e.,* an ad-hoc file-sharing network). Peers in a swarm regularly announce themselves to the central server coordinating their session, known as "tracker". A tracker has knowledge of all the peers that are part of the swarm it coordinates, and advertises this list among them as they announce themselves to the tracker, as shown in Figure 1.

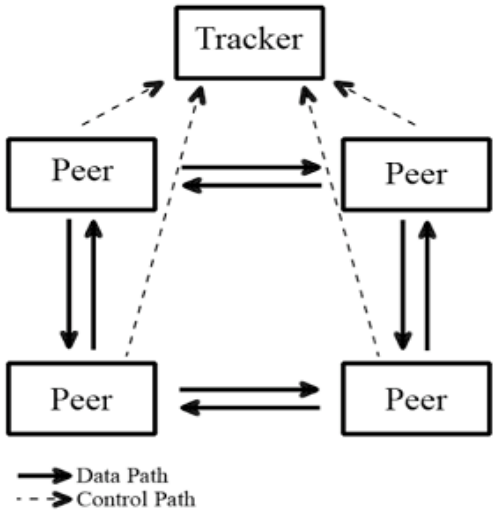

Figure 1: Architecture of a BitTorrent swarm [2]

For the file transfer to start, the person willing to distribute the original file must create a “torrent” file and host it in a server running tracker software. Among other things, this torrent file contains the URL address of the tracker to which peers should announce. The torrent also contains all checksums of the hashes computed for each chunk of the file to be distributed. Once the torrent is uploaded to a tracker server, the original distributor of the file can start “seeding” the file. That is, he can start a BitTorrent client and load the torrent file on it and announce periodically to the tracker software.

BitTorrentSim is an extension to version 2.29 of ns-2 that implements the BitTorrent protocol. It allows for a generalized study of BitTorrent protocols behavior as it does not focus on a particular version of BitTorrent. Additionally, it gives a desirable level of abstraction up to the OSI Application Layer which served to simplify our BitTorrent simulation study [1]. Currently, it is the only available open source implementation of BitTorrent for ns-2.

3. Approach

In our proposal, we mentioned that we were going to deploy a malicious tracker with a modified announce function. The idea was to change this function to return a modified peer list containing the IP address of an arbitrary host (preferably a host outside of the swarm) as a fake peer so the other peers in the swarm would try to connect to him.

However, it was not possible for us to modify the BitTorrentSim framework to support this attack with a target outside of the swarm, as this would require changes into a considerable part of the framework logic. For example, the tracker in BitTorrentSim can only keep track of instances of the class BitTorrentApp. Therefore, it cannot put a TCP agent or any other higher level TCP object as the target of attack. For this reason, we set the target as a node in the swarm, and changed the code of tracker and peers so that we can monitor the network traffic attack of a peer under attack.

Below we show a figure depicting our new approach, which differs from our original proposal with respect to the identity of the target of attack. Here, our target is part of the swarm, as shown below.

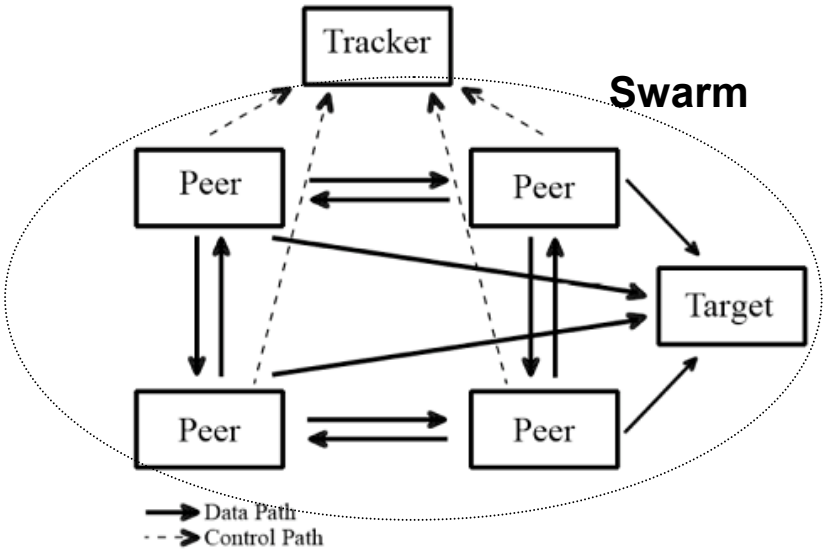

Figure 2: Architecture of a modified BitTorrent swarm with a malicious tracker [2]

Given these assumptions, we describe below the phases to evaluate the effectiveness of the attack.

3.1 Phase I: BitTorrent Simulator

In this first phase, we got the BitTorrentSim to work, which is an implementation of the BitTorrent protocol for the network simulator ns-2. First, we applied the patch to the version 2.29 of ns-2, so we were able to run BitTorrentSim with the ns-2 code provided we tried different scenarios. For running the simulation we needed to provide certain parameters:

```
      <Usage>: ns <tcl file> <Number of peers> <Non/deterministic>
<Max upload rate kb/s>
```

We use a *flashcrowd* star topology, with one seed and "Number of peers" minus one leechers. This topology is created on the .tcl file, and we vary the number of peers as this is the parameter that impacts the network attack traffic most significantly. The output provided by BitTorrentSim is a log file with a line for each peer. Each line contains the id of the peer, the time when this peer joins the swarm, the time at which it gets the first chunk, the time at which it finishes downloading the file, and the difference between the time when it ended and the time when it started the download. The first line in the the log file is for the seed, and it never receives any file data because it already has the file.

Analyzing the source code of BitTorrentSim, we concluded that the main C++ classes of this framework were BitTorrentApp, which represents an instance of a peer, and BitTorrentTracker, which is the tracker of the simulation. BitTorrentSim takes as input the number of peers and sets the first peer as the only seed for the entire simulation. After that, BitTorrentSim also creates one BitTorrent tracker. As the simulation goes, each peer announces itself to the tracker. When the tracker receives this announcement, it returns a random set of peers that have joined the swarm to the announcing peer.

3.2 Phase II: Incorporate Modifications into BitTorrentSim

In this phase, we discussed the different ways of measuring the incoming attack traffic on the target node. We agreed on measuring the network attack traffic as the TCP requests (both number of packets and kilobytes) received on the target node at fixed time intervals. Furthermore we noticed that as we increased the number of peers, the total time of simulation would considerably increase, and not necessarily reflect a notion of real time. Because timers in ns-2 are event driven and its virtual clock has some limitations, it is very hard to enforce real time constraints [6]. Therefore, we decided to calibrate our fixed time intervals according to the size of the swarm being simulated. That is, the larger the number of peers, the bigger the fixed time interval between each measure of attack traffic in the target node. In the following sections, we outline the main modifications. We performed on separate files of BitTorrentSim.

One of the first things we modified was the tcl file representing the flashcrowd topology. Basically, we enabled nam tracing so that we could visualize the traffic in the network. We made this option available as an extra parameter to the tcl file ("1" to enable nam tracing, "0" to disable it). Nevertheless, one of the problems we faced is that the trace file gets too large for a number of peers bigger than 10. This happens because the total number of packets exchanged between more than 10 BitTorrent nodes generates a large number of traces for nam. We concluded that

this simulation does not properly scale for having nam support a great number of peers in the swarm.

3.2.1 Modifications to C++ files in BitTorrentSim

- bittorrent_tracker.cc/bittorrent_tracker.h

This is the class representing a BitTorrent Tracker. At first, we tried modifying it to return the address of an arbitrary host (*e.g.,* TCP agent) on the simulation. However, because the logic of BitTorrentSim is so dependent on the values a BitTorrent node can return to the tracker, we decided to simulate the attack directed to an arbitrary peer in the swarm.

According to the attack in [2], a BitTorrent tracker should contain a function called announce. This function is the one activated when peers announce themselves to the tracker, and it is where the tracker returns a random set of IPs of the peers that are part of the swarm. In BitTorrentSim, this functionality is implemented in the method `get_peer_set(int req_num_of_peers)` from the BitTorrentTracker class. Therefore, we modified this method to always return the node address (equivalent to IP in ns-2) of the target of attack in the peer set, so that all the other peers in the swarm try to connect to and request file chunks from it. In addition, according to the authors of this attack, this node should always be placed in the first position of the returned peer set, as this ensures that peers will try to connect to it first [2]. Below we show the C++ code that forces the address of the target of attack to always be part of the set returned by the tracker, as well as to always place it as the first element in the set:

```
vector<Node*> BitTorrentTracker::get_peer_set(int req_num_of_peers) {

(...)

    // CS463 -> add TARGET OF ATTACK (first peer to join the swarm
    // after seed) to first position in the return set
    if (length > 1) {
        // CS463 -> insert first non-seed peer to join the swarm(target)
        return_set.insert(return_set.begin(), peer_ids_.begin()+1,
        peer_ids_.begin()+2);

        /* CS463: to enforce that this is only done once, we set the target
           of attack only when our return size has 2 elements.
           that is, the initial seed and the target of attack. it is also
           here that we can create the attack traffic statistics timer.
         */
        if (length == 2) {
            // CS463 -> set it as the target of attack
            BitTorrentApp * target = (*return_set.begin())->getBTApp();
            target->set_target_of_attack(true);

            // CS463 -> create and start thread
            ThreadClass *stats_timer = new ThreadClass();
            stats_timer->run(target);
        }

        /* CS463 -> remove duplicate. that is, skip first element and search
           for peer with same address as peer on top
         */
        for (it = return_set.begin()+1; it != return_set.end(); it++) {
            if ((*it)->address() == (*return_set.begin())->address()) {
```

```
                return_set.erase(it);
                break;
            }
        }
    }

    (...)

    return return_set;

} // end of get_peer_set
```

- bittorrent_app.cc/ bittorrent_app.h

This is the class representing a BitTorrent peer. Among other attributes, it has a pointer to an instance of the Node class, a TclObject from ns-2. This node object contains the address of the peer. It is essentially this address that the tracker handles to each peer inside the return set on `get_peer_set(int req_num_of_peers)` function.

Since we want to measure the network attack traffic as the amount of BitTorrent request messages received at the target of attack, we need a way to differentiate a "regular" peer from the peer that has been chosen as the target of attack. We do this by adding two extra functions in the BitTorrentApp class: `void set_target_of_attack(bool val)` and `bool get_target_of_attack()`. As the names of these functions suggest, they are used to set a regular peer as the target of attack, and to check whether a peer is the target or not. Given the above functions, we assume the first regular peer (*i.e.,* first one after the seeder) to join the swarm is the target of attack. Because the tracker is the first to know what peers have joined the swarm, it is he who sets the appropriate one to be the target of attack. timer_thread.cc/ timer_thread.h

We implemented this class to keep track of BitTorrent request messages received at the target peer, as BitTorrentSim did not allow this. The class implements a user controllable timer that measures the amount of network attack traffic as the number of request messages and number of kilobytes received at the target as a function of real time measured in seconds. We set the time interval for each measure in timer_thread.h, and we create only one timer thread for the duration of the whole experiment.

3.3 Phase III: BitTorrent DDoS experiment

In this phase, we decided to run the simulation and keep track of the number of connections made to the target of attack (the first peer on the list retrieved from the tracker), as well as the kilobytes received on that node (total and per time interval). We ran this simulation with different swarm sizes so we can notice the changes according to the parameters. In the next few sections, we describe our results.

In our proposal we discussed the possibility of experimenting in a real environment the impact on the availability based on the amount of unsolicited TCP traffic. However, this type of experiment became out of scope for our project, as these results are largely dependent on the type of system under evaluation, and cannot be generalized due to the variance of systems and networks. Therefore, this type of study would be best suited in a separate project.

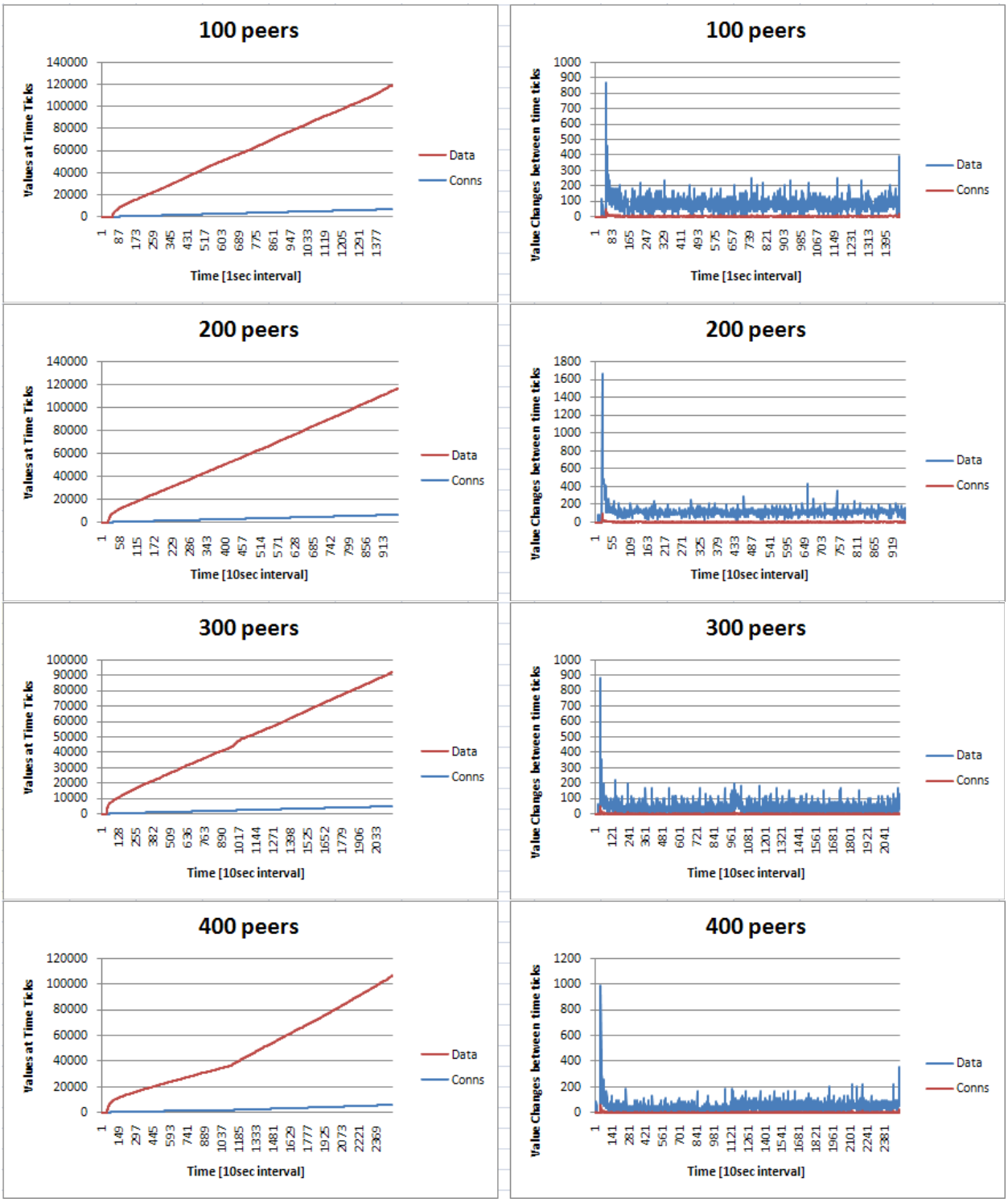

Figure 3: BitTorrent requests received at the target node.

We conducted our experiments on variations of different size of star topologies. Figure 3 shows our results for the star topology with 100, 200, 300 and 400 peers in the swarm with a maximum upload rate of 2048 kilobits/s. It is important to note that we measure BitTorrent request

packets, and we do not consider actual file chunks sent to the target, as file chunks would not have been sent in a real deployment of the attack.

The left graphs show the total amount of BitTorrent requests (as number of connections and kilobytes) that the target node received at every time slot. The amount of data and connections is increasing as time goes by as the peers who joined the swarm are requesting chunks of the file being downloaded.

In the right side, we can see in the graphs that the first peak is when all the peers are establishing TCP connections to the target. This peak reflects a major finding in our experiment. Such a large burst of traffic including so many connection requests can lead to a degradation of TCP network traffic at the target. It is shown how the lengths of sudden large amounts of DoS traffic (bursts) can worsen the throughput of TCP aggregate bandwidth and can cause the target to deny access to other connecting machines, even when the total amount of data is not that high [5].

Also in the right-side graphs, we can see that there are further sudden changes in the connection requests as well as the data requests, oftentimes double the average amount otherwise in the graph. While these peaks might not be enough to incapacitate the target host, it would lead to a substantial degraded TCP performance from these sudden bursts of traffic.

The effectiveness of the attack varies given the total amount of peers in each graph. For instance, the ratio of the peak value to the average traffic after that initial peak increases exponentially as function of the number of peers in the swarm, as can be seen in Figure 4:

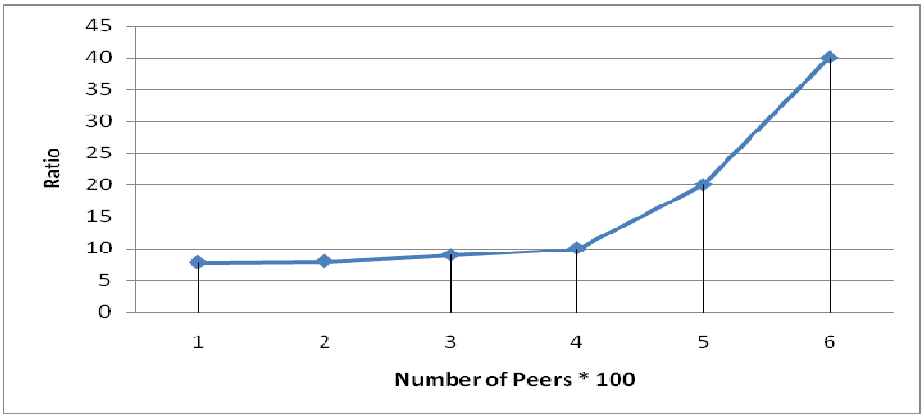

Figure 4: Ratio between initial peak value and average TCP requests sent to target.

Because the target peer is always present in the set of addresses returned by the tracker, the amount of data received as BitTorrent handshake increases as a function of the number of peers. In addition, the total amount of traffic coming from larger and larger sets of peers has an increasing trend. For instance, we can see that at in a swarm with 400 peers there is a substantial enough set of burst peaks to create a degrading effect on the target host.

Thus, we can conclude that it is possible to deploy a successful DoS attack with a BitTorrent network. However, this depends on a number of factors:

1) *Whether one has control over a BitTorrent tracker.*

There exist myriad trackers on the internet with unidentified owners which make this quite easily the case. Furthermore, trackers are usually used for illegal purposes so swarm users are limited in their recourse against the tracker.

2) *The number of peers that are part of the swarm.*

   In general, according to our experiment, this attack will increase the amount of unsolicited traffic at the host. The amount of traffic actually given to the target may increase enough as the size of the swarm becomes larger to cause a serious denial of service, but even with smaller swarm sizes, it is shown that there is enough traffic to cause a substantial degradation of service at the target host [5].

3) *The popularity of the torrent file.*

   We estimated that, in average, the most popular torrent files on the internet [7] have at least 10.000 peers in the swarm.

According to our data, given a maximum upload rate of 2048 kilobits/s, our simulation shows that a normal size BitTorrent network can generate a considerable amount of TCP handshake traffic to a single host. This initial peak value, as our data shows, increases exponentially as a function of the number of peers. These results were obtained with the target of attack simulated as a peer that was part of the swarm. We, however, conjecture that if instead the target of attack was a host outside of the swarm, *there would be additional TCP handshake peaks*. That would happen because the target of attack would not establish a BitTorrent connection. Because TCP does not know if the packets were received at the target, and because BitTorrent clients fully trust the tracker to send valid peers and do not mark a non-responding peer as invalid, the peers would retry the connection, and cause further peaks.

Therefore, the effectiveness of this type of denial of service attack varies according to the above-mentioned factors. Given that it cannot be easily detected in a network that does not filter BitTorrent traffic and that peers in the swarm currently do not have any detection mechanism to tell if they are part of an attack network, this attack could effectively cause a degradation of the target's network services availability up to the point of a DoS. The effect is particularly dramatic given the increasing popularity of the BitTorrent protocol.

References

[1] Eger, K., Hoßfeld, T., Binzenhöfer, A., and Kunzmann, G. 2007. Efficient simulation of large-scale p2p networks: packet-level vs. flow-level simulations. In *Proceedings of the Second Workshop on Use of P2p, GRID and Agents For the Development of Content Networks* (Monterey, California, USA, June 25 - 25, 2007). UPGRADE '07. ACM, New York, NY, 9-16. (http://www.tu-harburg.de/et6/research/bittorrentsim/index.html)

[2] Harrington, J., Kuwanoe, C., Zou, C. C., A BitTorrent-Driven Distributed Denial-of-Service Attack. *3rd International Conference on Security and Privacy in Communication Networks (SecureComm 2007)*, Nice, France, September 17-20, 2007.

[3] El Defrawy, K., Gjoka, M., and Markopoulou, A. 2007. BotTorrent: misusing BitTorrent to launch DDoS attacks. In *Proceedings of the 3rd USENIX Workshop on Steps To Reducing Unwanted Traffic on the internet* (Santa Clara, CA). S. M. Bellovin, Ed. USENIX Association, Berkeley, CA, 1-6.

[4] Cohen B., “BitTorrent Protocol Specification”, http://www.bittorrent.org

[5] Kuzmanovic, A. and Knightly, E. W. 2006. Low-rate TCP-targeted denial of service attacks and counter strategies. *IEEE/ACM Trans. Netw.* 14, 4 (Aug. 2006), 683-696.

[6] Mahrenholz, D. and Ivanov, S. 2004. Real-Time Network Emulation with ns-2. In *Proceedings of the 8th IEEE international Symposium on Distributed Simulation and Real-Time Applications* (October 21 - 23, 2004). Distributed Simulation and Real-Time Application. IEEE Computer Society, Washington, DC, 29-36.

[7] The TorrentZ torrents search engine, http://www.torrentz.com