// Scientific News of NTUU "KPI".—K.: 2008, N 35.—P.123-129.

УДК 004.75, 004.724.2

THE POSSIBILITY OF REPORTED TRAFFIC FORGERY ON PRIVATE BITTORRENT TRACKERS

Poryev G.V., Poryev V.A., National Technical University of Ukraine "KPI", Kiev, Ukraine

The overview of the development of concept of peer-to-peer networks is given while historical trends of its progress are analyzed. The content's lifecycle and load-balancing techniques in BitTorrent networks are reviewed. It is shown that the traffic reports on private trackers could easily be forged.

Introduction

The concept of peer-to-peer networks, not nearly new at the beginning of XXI century, was briefly outlined in the times of Internet very inception back in 1969. Although the contributors in ARPA could not possibly have predicted the future scale of worldwide distribution of what was then a single link between just two mainframe computers, the idea of interconnected peer nodes was already there.

It should be noted though, user interface terminals at the time were nowhere near to compare with host computers (mainframes), and were essentially lacking any computing and storage facilities whatsoever, hence the vision of peering networks remained dormant for long time since.

Only as the mainstream computers surged into the consumer market during 1970s and 1980s, the legacy of what we know today as "client-server architecture" was to be dominant for decades to come. It was assumed that should there be a network, it is naturally divided into servers (that provide access to resources) and clients (that make use of provided resources). The performance and capacity gap between server and client hardware and, which is more important, a difference between network interconnections was still too obvious.

At that time, peering was common practice when dealing with server software and network architecture. TCP/IP routing schemes was essentially peering to the point that the very word "peering" made it into the specific technical term on internetworking routing, despite the fact that actual physical channels had (and still have) visible relevance to national backbones and traffic exchange points, making them more or less subordinate to each other. However, Usenet and e-mail servers were communicating with each other and there were no such thing as primary layer or central hub(s) through which all traffic should be passed — which is peering network.

Outside of Internet, attempts to build peering networks were also undertaken. One of the most successful of those attempts was FidoNet — amateur worldwide computer network, initially consisting of independent bulleting board systems (BBS), built on packet-switching principle over regular telephone lines using dialup modems. Unlike Internet, FidoNet is not online-network and all user interaction could be and mostly done in offline state. Host software, however, is required to maintain online availability during the certain policy-defined hours each day.

Right upon emerging, the FidoNet was truly peering, in the sense that each originating node accessed its addressee directly by calling its address (phone numbers in this case). Later in 1990-x, however, FidoNet had also "suffered" from infrastructure growth, when the network had exploded into thousands of nodes worldwide. These times of FidoNet development were marked with strict hierarchical structure, roughly based on geography and various regulating authorities within the network. It is worth noting, that unlike Internet (IPv4 address space making up 2^{32} addresses, including non-routable and reserved), hierarchical address structure of FidoNet theoretically allowed address space of 2^{48} network nodes alone and 2^{64} connection points in total.

Despite all aforementioned advances and peeks into the future concept, truly peer-to-peer online networks as we understand them today were far from reach before the advent of third millennium.

The commercial grounds for real peer-to-peer networks have appeared not until permanent Internet connections (also called then "leased lines") built on technologies such as ADSL or DOCSIS gained significant consumer market at homes and offices. In addition, not until average home and office computer hardware was closing to the average server hardware (often being built from the same parts indeed) was it plausible to build peer-to-peer networks with evenly distributed computing and storage resources [1].

It is widely believed, that commercial applications of the concept started to appear and gained much popularity in the beginning of XXI century.

Overview of BitTorrent

One of the modern peer-to-peer network protocols, BitTorrent, was conceived in 2001 and to date remains responsible for largest part of consumer-generated Internet traffic, sometimes prompting Internet Service Providers (ISPs) to implement special, often unpopular, filtering measures and devices.

Unlike other popular peer-to-peer networks such as eDonkey2000 or Gnutella networks, BitTorrent does not constitute a single addressing or naming space. It is not even a network itself, because BitTorrent operates as multitude of independent content-tracking servers, called "trackers". Each tracker maintains the list of published content entities, and for each entity, it maintains the list of peers associated with it. Most trackers do not communicate with each other, as eDonkey2000 servers do, unless they are sharing same content and are specially designed to exchange information among themselves.

Due to the absence of overhead related to maintaining global naming or addressing space, BitTorrent networks are quite faster in comparison with eDonkey2000 or Gnutella in terms of download and upload speed and length of download queues. BitTorrent clients are most likely to consume their bandwidth to exhaustion, despite the fact that BitTorrent does not imply sophisticated load-balancing algorithms for upload, reward scores and so on [2].

Typical content lifecycle in BitTorrent could be described as the following:

1. **Preparation** — content publisher prepares torrent file, which describes the number, names and size of files and the control checksums of each slice of binary stream made up from content files.
2. **Publication** — publisher uploads torrent file in such a way that tracker became aware of its existence, not necessarily knowing all the details specified in the torrent file.
3. **Distribution** — publisher distributes torrent file among clients who wish to download its content. It is usually done through web-based forums, either public or private or via other means. It is worth noting that publication and distribution is not the same process, although in most cases they are done simultaneously in the scope of one server. For example, uploading torrent file as file attach to the message on forum automatically registers torrent contents in the tracker.
4. **Initial seeding** — publisher running BitTorrent-compliant client starts accepting incoming requests for content.
5. **Leeching** — other clients proceed to download published torrent file, requesting tracker for the address of initial seeder and requesting initial seeder for content.
6. **Downloading** — clients actively downloading content file will enable already downloaded slices to be shared among other clients, effectively speeding up the transfer for them.
7. **Secondary seeding** — clients that completed the download, engage in seeding it by themselves.
8. **End of interest —** all involved clients finishes and became seeders, and no downloading clients are left in the swarm.
9. **Fadeout** — seeders stop seeding one by one, and eventually there are neither seeders nor downloading clients associated with this torrent.

Once the content entity is fully downloaded (the transition between stage 6 and 7), the BitTorrent client must ascertain the data integrity of it. In this part BitTorrent specification seems to be slightly under-developed in comparison with its counterparts of eDonkey2000 and Gnutella networks. While the latter does use sophisticated tree-hashing algorithms designed to minimize traffic overhead, BitTorrent simply calculates hashing stream from binary stream with variable-sized chunks. If an error is detected, the whole chunk needs to be re-downloaded.

Analysis of Load-Balancing technique

Most peer-to-peer network will eventually encounter the phenomenon called “leeching”. The network client involving in leeching will only download content and not share it among others. Although such behavior is necessary for some time just after initial publication of the content (since some time is required to download at least one complete shareable piece of data), leeching beyond necessary period and for long time is considered bad, because it forces excess resource usage on other clients interested in the same content [3].

Peer-to-peer networks often employ various sophisticated algorithms to discourage leeching.

One of prominent example is the credit reward system found on popular eDonkey2000 clients. Such clients maintain a “performance record” for each incoming client, who expressed interest in published content.

Typically, incoming clients are arranged into queue in order of time of their appearance. The foremost client in queue is served by the content piece and then rescheduled at the end of queue, therefore advancing other queue members.

However, incoming client can advance queue member by more than single step in the queue, taking into account its contribution (in case the sharing client is not completed seeder, of course). That is, the more content pieces were provided by the incoming client, the faster it progresses in the queue. This effectively places “bad” leechers to the end of queue and slows their advance.

Unfortunately, no such reward system is currently employed by the majority of the BitTorrent clients. There are number of reasons for it, including the aforementioned difference in distribution speed (BitTorrent content usually distributes faster than comparable eDonkey2000 counterpart due to small size of swarm). However, similar scheme are designed in so called “private trackers”.

As BitTorrent is developing technology, new protocol extensions are constantly added to improve the overall efficiency of content sharing. These include, for example, so-called “Fast Peer Extensions” to allow new peers bootstrap into swarm more rapidly. Although it is uncertain whether the performance itself is nearly topping its potential for the current BitTorrent development stage, it is beyond the scope of this paper.

Public vs. Private Trackers

Roughly, trackers can be called “public” or “private”. Public tracker, such as famous Sweden-based ThePirateBay usually does not require invitation or registration to be able to download its advertised content, therefore do not maintain download and upload rating records of its users.

In contrary, private trackers, such as Torrents.Ru, do implement some restrictions against anonymous access. This is possible using so-called private keys — special passwords attached to the announce URL of tracker, designed so that tracker could ascertain the user identity of every announce or update request coming from BitTorrent clients.

Private trackers often employ rating system, where rating is a value calculated using various formulas including overall download and overall upload amount of a particular user. Users with low rating are restricted from further downloading or they are potential candidates to be banned from tracker. Users with high rating have certain privileges such as ability to download more torrents simultaneously, priority to access and search across tracker, etc.

Hence in order to encourage content sharing and discourage leeching, tracker server must somehow be made aware of how much some particular BitTorrent client did download and upload to others. This is currently made by issuing special HTTP request (“tracker updates”) to the tracker. Such requests usually contain user identity, content identity (hash), client activity state, amount of downloaded and uploaded data and other relevant information [4].

Analysis of Tracker Update requests

Since BitTorrent specification is open to the public community, it is known that any part of mentioned information request could be forged or faked, and therefore used to “illegally” boost the user rating. However, because this would require significant level of software engineering knowledge, the fact is not widely known.

Let us consider an example of tracker update (long lines were split for the reader’s convenience):

```
GET /announce.php?uk=3b02d5XTYZ&
&info_hash=%da%5d%a4H%20%e6%d23%25%cag%b9%10x%3f.%a0%ffk
%e9
&peer_id=-UT1750-%fa%91%07%e1%10n%c3O%96%d1%be%3b
&port=8080
&uploaded=311181312
&downloaded=0
&left=0
&key=8AA14C62
&numwant=200
&compact=1
&no_peer_id=1
HTTP/1.1
Host: bt.torrents.ru
User-Agent: uTorrent/1750
Accept-Encoding: gzip
```

The most significant parts of tracker update are described below:

uk — stands for “user key”, identifies user within private tracker.
info_hash — primary hash for torrent content.
peer_id — client software id (UT1750 means μTorrent 1.7.5.0).
port — port at which client software accepts incoming requests.
uploaded — amount of data uploaded in the scope of this torrent.
downloaded — amount of data downloaded in the scope of this torrent.
left — amount of data required to complete torrent.

We see, that amount of data downloaded and uploaded are reported to private tracker server by the client. Tracker server cannot validate this claim directly, because it does not know exactly, which peer is downloading or uploading to which peer in the swarm. Even indirect calculations, based on the estimation of number of leeching, downloading, and finished clients and overall traffic they reported, could be dramatically wrong because some clients might have cancelled downloads or there may have been communication issues between tracker and clients, but not among clients themselves, as often happens in large home networks.

Conclusion and recommendation

There are number of ways in which reported “downloaded” and “uploaded” values could be faked. Most flexible but terribly time-consuming process would involve development of own BitTorrent client or modifying existing open-source client. Other way, sometimes described in hacking-related forums on the Internet, would require interfering with active connection between client and tracker, suppressing legit tracker update request and inserting forged one in its place. This too, however, requires deep understanding of TCP/IP implementation and reverse-engineering skills.

We hope that future generations of BitTorrent protocol will employ sophisticated methods to block this type of “attack”.

References

1. Stephanos Androutsellis-Theotokis, Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies // ACM Computing Surveys,—2004.—36(4).—P.335–371.
2. Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. Technical Report UW-CSE-01-06-02, University of Washington, Department of Computer Science and Engineering, July 2001.
3. Poryev G.V. The Application of the Peer-to-Peer Network Technologies // Proceedings of Scientific Workshop of Donetsk National Technical University. Issue #12(118) “Computing Technology and Automation”.—DNTU, Donetsk (Ukraine), 2007.—p.150.
4. Poryev G.V. Data Integrity Control in the Distributed Networks // Western-European Magazine on Advanced Technologies. Issue #4/2(22).—KNURE, Kharkiv (Ukraine), 2006.—P.32-35.

Порев Геннадий Владимирович, Порев Владимир Андреевич Возможность подделки отчётов трафика в частных трекерах BitTorrent Выполнен краткий обзор развития пиринговых сетей и его исторических тенденций. Проанализирован жизненный цикл контента и способы распределения нагрузки в сетях BitTorrent. Показано, что отчёты	Порєв Геннадій Володимирович, Порєв Володимир Андрійович Можливість підробки звітів трафіку в приватних трекерах BitTorrent Виконано короткий огляд розвитку пірінгових мереж та його історичних тенденцій. Проаналізовано життєвий цикл контенту та способи розподілу навантаження в мережах BitTorrent. Показано, що звіти трафіку в

трафика в частных трекерах BitTorrent могут быть подделаны.	приватних трекерах BitTorrent можуть бути підроблені.