P2P proxy/cache

A hybrid P2P/CDN networking approach eliminating the problems of P2P through caching/relaying trackerless BitTorrent traffic

Ivan Klimek

Computer Networks Laboratory,
Department of Computers and Informatics,
Technical University of Košice, Letná 9, 041 20 Košice,
Slovak Republic.
Tel: +421-902-152873
E-mail: Ivan.Klimek@cnl.sk

ABSTRACT

This paper describes the addition of trackerless torrent caching to our existing P2P proxy cache. Trackerless torrents represent the next evolutionary step of the BitTorrent protocol; they eliminate the need for centralized trackers which in fact are the weak point of the whole technology. Further, it will be shown that our approach enables total user anonymity. This two mentioned features together with the massive reduction of traffic behind the proxy achieved through avoiding redundancy, solves practically all the problems of P2P without the need to change the used technology, but just add features transparently above it. The motivations behind caching of P2P traffic won't be described in this paper as there were already studied deeply [1].

1 TRACKERLESS TORRENTS

The original BitTorrent protocol was not completely decentralized; it relied purely on the centralized control servers named trackers for coordination of the peer cloud. These trackers represented a single point of failure; their take down would render the whole technology useless. Also, this trackers have to be run by someone, this person(s) are exposed to possible legal actions against them even that the tracker itself doesn't hold any illegal content [2]. Because of these factors, the need to develop a decentralized alternative to trackers arose. Currently, there are three "trackerless" peer-discovery technologies being used:

- Distributed Hash Table (DHT)
- Peer Exchange (PEX)
- Local peer discovery

These trackerless peer discovery methods were not primarily developed to fully replace trackers, but just to add more peers resp. represent a fallback option.

1.1 Distributed Hash Table

Distributed hash tables (DHTs) are a class of decentralized distributed systems that provide a lookup service similar to a hash table: (key, value) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows DHTs to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures. [3]
Because we are focusing on the BitTorrent protocol, we will specify its DHT implementation:
Kademlia is a distributed hash table for decentralized peer to peer computer networks designed by Petar Maymounkov and David Mazières [4]. It specifies the structure of the network and the exchange of information through node lookups. Kademlia nodes communicate among themselves using UDP. A virtual or overlay network is formed by the participant nodes. Each node is identified by a number or node ID. The node ID serves not only as identification, but the Kademlia algorithm uses the node ID to locate values (usually file hashes or keywords). In fact, the node ID provides a direct map to file hashes and that node stores information on where to obtain the file or resource. When searching for some value, the algorithm needs to know the associated key and explores the network in several steps. Each step will find nodes that are closer to the key until the contacted node returns the value or no more closer nodes are

found. This is very efficient: Like many other DHTs, Kademlia contacts only O(log(n)) nodes during the search out of a total of n nodes in the system. Further advantages are found particularly in the decentralized structure, which clearly increases the resistance against a denial of service attack. Even if a whole set of nodes is flooded, this will have limited effect on network availability, which will recover itself by knitting the network around these "holes". [5]

The BitTorrent DHT specification [6] mentions that instead of using trackers in the .torrent file a peer can be specified. This peer then supplies a list of other active peers and by that replaces the function of a tracker. In fact, this is replacing a single point of failure with another single point of failure. Further, it looks like [7] a default peer that is hardcoded in the client is always contacted even on torrents with a specified tracker. In uTorrent and in the mainline BitTorrent client it is router.bittorrent.com (this one is also mentioned in the official DHT specification) or router.utorrent.com respectively. Because BitTorrent is a commercial company, it cannot be guaranteed that filtering of content resp. legal actions against users won't occur.

1.2 Peer Exchange

Peer exchange (PEX) is a feature of the BitTorrent peer-to-peer protocol which, like trackers and DHT, can be utilized to gather peers. Using peer exchange, an existing peer is used to trade the information required to find and connect to additional peers. While it may improve (local) performance and robustness—e.g. if a tracker is slow or even down—heavy reliance on PEX can lead to the formation of groups of peers who tend to only share information with each other, which may yield slow propagation of data through the network, due to few peers sending information to those outside the group they are in. For "trackerless" torrents, it is not clear if PEX provides any value since the mainline DHT can distribute load as necessary. Each DHT node acting as a tracker may store only a subset of the peers, but these are maximal subsets constrained only by DHT node load rather than by a single peer's view. Private torrents disable the DHT, and for this case, PEX might be useful provided the peer obtains enough peers from the tracker. [8]

PEX like DHT needs an existing peer to gather other nodes to connect to. Although, there is no "default" peer like in DHT.

1.3 Local Peer Discovery

A peer with enabled Local peer discovery sends multicast messages, if there is another peer in the same multicast domain and it has the content identified by the infohash in the multicasted request it will reply to the sender. This mechanism works only on local segments as multicasts are usually filtered on the gateways, also speed limits do not apply on transfers between hosts discovered using Local Peer Discovery.

2 TRACKERLESS TORRENT CACHING

2.1 DHT caching

Because of the protocol design and its usage of UDP, it is simple to detect and initialize a Man-in-the-Middle attack on DHT. The messages are always in the same format so the methods developed for intercepting HTTP tracker requests can be used out of the box. [1] This is also true when the protocol encryption is used, as it does not encrypt the DHT initialization messages.[1]

2.2 Peer exchange

PEX does not work without knowledge of some "prior" peer. With the control over DHT there is no reason why we should focus on it.

2.3 Local peer discovery

If the proxy cache will be placed on the same multicast domain as the clients, it is the easiest way how to publish the content. It just needs to listen for the multicasted requests.

3 AVOIDING MONITORING

BitTorrent is by no means an anonymous protocol, there are at least three ways how it is possible to identify what is the user downloading:

1) Every peer gets a list of other peers to which it then tries to connect
2) The tracker knows all the peers and what are they downloading
3) Eavesdropping on the network communication - BitTorrent communicates mostly in clear text, even with the protocol encryption turned on it is possible to determine who is downloading what because the protocol encryption was designed to obfuscate protocol recognition mechanism not to protect privacy.

1 This is primarely for backwards compatibility reasons.

P2P proxy cache is able to defeat all this methods and guarantee almost full anonymity without the need to modify the protocol thus existing client SW can be used.
(Full anonymity is also possible by minor additions)

1) In a network served by a proxy cache, the only visible peer is the proxy cache itself.

2) The original client's request never reaches the tracker, the same is true for the mentioned DHT "default peers".

3) With the proxy cache deployment client's traffic stays in the original ISP's network, e.g. only few hops to the nearest proxy cache. This massively reduces the chances for eavesdropping - which would need to be done directly by the peer's ISP. We will present a solution to make this bulletproof later too.

4 LEGAL ISSUES

Peer-to-Peer networks clearly represent a legal issue, numerous trials with the users of P2P systems or people behind the BitTorrent trackers are everyday in the news.

Let's summarize the facts:

- Caching content as defined by the DMCA [9] is not illegal, even storing and relaying copyrighted content for the purposes of caching is not illegal [10].
- The article 6 of IPRED[2] gives the power to the interested party to apply for evidence regarding an infringement that lies in the hands of the other party to be presented. The only requirement is for that party to present "reasonably available evidence sufficient to support its claim" to courts. [11]

Thus, according to our interpretation using caching as described it is possible to legally defeat current monitoring mechanism[3] and by that protect P2P users as their activity can no longer be monitored.

2 Directive 2004/48/EC of the European Parliament and of the Council of 29 April 2004 on the enforcement of intellectual property rights

3 The author is not aware of any monitoring mechanism that could be used to gather sensitive information on users when caching as presented would be applied.

5 CREATING AN ANONYMOUS NETWORK

Trackerless torrents represent a great progress for the whole protocol, but they are limited in ways described earlier. To enable them fully replace trackers and become more decentralized/secure, the P2P proxy cache would need to be deployed in larger scale and create a defacto Cached Content Delivery Network (CCDN) like the Coral CDN [12]. This would enable to create a set of almost nonstop available peers, which could share a common DHT table which would be enlarged with every new download. These nodes would be then used instead of the default DHT peers (mentioned earlier). It is logical that a point would come where people would start to add this nodes to their torrents as the default peers, this could be done using a dynamic DNS entry pointing to the nearest most optimal proxy cache for the given peer. This can but does not need to happen, without it the CCDN would function completely transparently. Further, when the CCDN would become unavailable, for whatever reasons, the original default peer can be still accessible creating another fallback option.

With such acceptance/deployment of proxy caches, trackers would become obsolete. Their only role would be searching and holding the ".torrent" files. Both of these functions can be omitted with minor additions to the DHT protocol - so that it would support direct search in the DHT table/cloud. [13]

Caching is not limited to be retro-active[4], but it can be also pro-active. Cache every new content, only when it is not popular delete it. This would enable content to be shared much faster to great masses. This idea is not technically problematic, as at every moment the active amount of content represents only a fraction of the whole content.[5] It is only a question of setting the limit of what is active / inactive and the available disk space. Also pro-active approach would avoid completely any redundancy which is the key factor for caching.

Similarity Enhanced Transfer in its easiest form can be used to avoid further redundancy that is

4 Caching content after it becomes popular.

5 Lets call a active torrent only a torrent that has at least one seeder and leacher. In that case, only about 100 000 torrents of 925 914 registered on "tracker.thepiratebay.org" is active. That means active content represents about 1/10 of all content provided.[14]

caused by the same content (or same pieces of content) in different torrents. We say "in its easiest form" because to avoid adding complexity to the proxy cache code we aim to use only the SHA checksums of pieces and their relative order to detect the same content in different torrents. This in fact is enough information to detect duplicate content even if SHA collision can occur [15], because the longer the correct order of SHA hashes the bigger the probability of having discovered duplicate content.

In some jurisdiction it may one day be necessary to provide logs for content being cached e.g. who provided the cache with what content. This may well happen to stop sharing illegal content. The proxy cache has to be able to support this request and provide the logs.

A possible scenario:

- A Law enforcement organization send a request to take down content with a given infohash
- P2P proxy cache (all participating in the CCDN) take down the given infohash from its (their) database, so no new download of that content can occur, active downloads will continue, as it is technically impossible to stop them
- because P2P proxy cache uses the ZFS file system[6], the data are there, just the link in the database pointing to them was deleted
- the original seeder - "author" of the content realizes that the content is no longer in cache (this can be automated via a robot) and recreates the torrent, because the creation date/time is different from the original torrent the infohash will be different even that the data are the same
- the search engines (e.g. trackers) update their databases[7]
- as P2P proxy cache is a pro-active cache it detects new content and starts to download it, it detects it is the same content using our SET implementation and recovers the needed blocks from the ZFS thus almost no downloading from the original seeder is needed
- the content is available again; when automated this procedure can take only a few seconds - no one will even notice and the proxy cache has done what was requested by the law enforcement agency[8]

P2P proxy cache is an automated system and it simply forwards the responsibility to the author - initial seeder of the torrent which can be connected using a secure connection (IPREDator [17] for example). It does protect ordinary people, keeps all the content always available, but still cooperates with all the legal authorities.

6 OTHER POSSIBLE ADDITIONS

As already mentioned, even when using a proxy cache, the user's traffic will need to get to the proxy cache through several hops in the ISP's network, which creates a possibility for eavesdropping. A possible solution would be to implement an SSL tunnel between the peer and the proxy cache. This, however, would require the modification of the client software. We plan to modificate an open source client and make the changes publicly available. Other possible approach would be to create a "loader" that could be started before the client software and create a tunnel to be used by the unmodified client. For example an limited VPN to the proxy cache, limited to BitTorrent traffic only. This would enable total anonymity with existing SW stack. However, by doing that the proxy cache would loose its transparency. A possible better approach would be to support implementing Friend-to-Friend (F2F) [18] features into the BitTorrent protocol. After their implementation into mainstream clients, the user could simply add the CCDN into his friend list.

In networks where uploading is not an issue - like cable operators, or FTTH the proxy cache can operate like a relay to enhance the speed of content delivery in the local segment. This means, all the anonymity features would still be used but the peers would actively participate in content delivery thus keeping the basic idea of P2P networks alive. This would be very effective because it is lot easier and cost effective to pass through traffic than to read it from any kind of storage medium.

6 ZFS supports natively the multi-layer storage system that P2P proxy cache needs to use [1], further ZFS does NOT delete any blocks, it always writes on new blocks.

7 This means the .torrent files have to be used instantly after being downloaded else they can become obsolete.

8 Trackers that accepted to filter content - like Mininova [16] have survived longer and with less trouble than trackers that ignored/refused such requests.

A generally accepted rule of thumb is that 1 hertz of CPU processing is required to send or receive 1 bit of TCP/IP [19]. For example 5 Gbit/s (625 MB/s) of network traffic requires 5 GHz of CPU Processing. This implies that 2 entire cores of a 2.5 GHz multi-core processor will be required to handle the TCP/IP processing associated with 5 Gbit/s of TCP/IP traffic. Since Ethernet (10Ge in this example) is bidirectional it is possible to send and receive 10 Gbit/s (for an aggregate throughput of 20 Gbit/s). Using the 1 Hz/ bit rule this equates to 8 - 2.5 GHz cores. (Few if any current day servers have a requirement to move 10 Gbit/s in both directions but not so long ago 1 Gbit/s full duplex was thought to be more than enough bandwidth.) Many of the CPU cycles used for TCP/IP processing are "freed up" by TCP/IP offload and may be used by the CPU (usually a server CPU) to perform other tasks such a file system processing (in a file server) or indexing (in a backup media server). In other words, a server with TCP/IP offload can do more server work than a server without TCP/IP Offload NICs. [20]

Using such technique would enable the relaying of data in tens of gigabits per second spectrum. Caching would be used only to avoid downloading the same content from outside of the served network thus avoiding redundancy outside of the given segment.

One day in the future, trackers won't be no more; technically the switch to a completely decentralized search is possible to do right now. The problem are the people, the process from what they know and use every day to searching directly from the client software has to be gentle. An example can be the Tribbler client [21]

7 CONCLUSION

This paper presented a few approaches that could possibly legally avoid current monitoring techniques and by that increase the Internet privacy. Caching of P2P traffic is an emerging technology, that if could integrate privacy features, would be beneficial not only to Internet Service Providers but to their customers too. We think that it is important to transparently update the already used protocols and not to design and try to create something new. That is why the P2P proxy cache works completely transparently on top of the BitTorrent protocol.

References

[1] I. Klimek. P2P proxy/cache. Computer Networks Laboratory, Technical University Kosice, December 2008.

[2] Wikipedia, "The Pirate Bay Trial", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/The_Pirate_Bay_trial

[3] Wikipedia, "Distributed hash table", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Distributed_hash_table

[4] CSAIL MIT, "Kademlia: A Peer to peer information system based on the XOR Metric", 2002, [Online; accessed 12-June-2009], [Online], Available: http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[5] Wikipedia, "Kademlia", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Kademlia

[6] BitTorrent.org, "DHT Protocol", 2008, [Online; accessed 12-June-2009], [Online], Available: http://www.bittorrent.org/beps/bep_0005.html

[7] BitTorrent.com, "Troubleshooting", 2009, [Online; accessed 12-June-2009], [Online], Available: http://www.bittorrent.com/btusers/guides/bittorrent-user-manual/faq-frequently-asked-questions/troubleshooting

[8] Wikipedia, "Peer exchange", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Peer_exchange

[9] BitLaw, "17 USC 512, Limitations on liability relating to material online", 2005, [Online; accessed 12-June-2009], [Online], Available: http://www.bitlaw.com/source/17usc/512.html

[10] Wikipedia, "Field v. Google", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Field_v._Google

[11] Europa.eu, "DIRECTIVE 2004/48/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 29 April 2004 on the enforcement of intellectual property rights", 2004, [Online; accessed 12-June-2009], [Online], Available: http://eur-lex.europa.eu/pri/en/oj/dat/2004/l_195/l_19520040602en00160025.pdf

[12] Wikipedia, "Coral Content Distribution Network", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Coral_Content_Distribution_Network

[13] Wikipedia, "Kad network", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Kad_network

[14] Torrentz.com, "http://tracker.thepiratebay.org/announce 100,000 - 100,050 of 925 914", 2009, [Online; accessed 12-June-2009], [Online], Available:

http://www.torrentz.com/tracker_258334187520&p=2000
[15] Wikipedia, "SHA hash functions", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/SHA_1
[16] Wikipedia, "Mininova", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Mininova
[17] Wikipedia, "IPREDator", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/IPREDator
[18] Wikipedia, "Friend-to-friend", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Friend-to-friend
[19] A. Foong, T. Huff, H. Hum, J. Patwardhan, G. Regnier. TCP Performance re-visited. Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS) March 2003, Austin, TX
[20] Wikipedia, "TCP Offload Engine", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/TCP_Offload_Engine
[21] Wikipedia, "Tribler", 2009, [Online; accessed 12-June-2009], [Online], Available: http://en.wikipedia.org/wiki/Tribler