# Search request routing in Bittorrent and other P2P based file sharing networks

© Scherbakov Konstantin

Saint-Petersburg State University
deflector@gmail.com

## Abstract

Existing p2p file sharing networks doesn't always give their users abilities to make an effective and fast searches for particular data. In this paper we introduce some improvements for main subset of these networks, especially for bittorrent and descript some well known methods of search query routing, used in other networks that may be useful for our purposes too. We describe some ideas of modification source of engines, used by usual and super peers of bittorrent network and illustrate some possibly useful changes in bittorrent *dht* structure.

## 1 Introduction

The utilization of internet bandwidth by p2p traffic grows very fast in the past years. The pure p2p system is presented by some number of nodes with absolutely identical functionality and connected between each other. Such type of distributed systems is a base of many file sharing networks like kademlia, gnutella2, etc. But there are a number of file sharing networks with not pure p2p structure, which are also called p2p networks. One of the most popular of them is bittorrent [1]. All these networks give their users abilities to share some files between each other and to find them and their source nodes in the network. Effective speed of file downloading in such type of networks depends on different factors including number of simultaneously running data search queries.

Many of p2p systems (especially file sharing) are unstructured: their nodes with shared data on it appear and disappear constantly, network topology changes in time. So, success of particular data search request depends on this feature.

There are some known and successfully implemented search algorithms in p2p networks. They can be united in some classes: blind and informed [19].

**Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2008**

### 1.1 Blind search methods:

- original gnutella algorithm (FS/BFS): one node contacts each other accessible nodes within TTL hop; produces huge overhead [4]
- Modified BFS (MBFS): each node send request to a random subset of other nodes, connected to it; number of requests reduced in comparison with FS [6]
- Iterative Deeping (ID): rather similar to previous two methods, but realizes a special feature – user defined termination condition; ID is implemented in ed2k/kademlia [8, 20]
- Random Walks (RW) [8]: initial node sends some number of search requests to different neighbor nodes, each other node processes this request (also known as walker) by itself and then resends it to only one neighbor; walkers terminates by success or failure (based on TTL or other walkers results)

### 1.2 Informed search methods:

- Super Peer (SP): initial (leaf) node contacts a special super peer (hub) node, when sending a search request; hub node processes this request and send it to relevant leaves and neighbor hubs [14]; implemented in gnutella2
- Intelligent BFS (IBFS): each node stores a special local index - pairs (query, neighbor_id) for recently processed queries; when a new request arrives, node checks its similarity to stored requests and then extracts target nodes, which returned maximum number of results to this requests and at least redirects new request to them, waiting for feedback with number of hits to update its local index [6]
- APS: each node stores a special local index – (neighbor_id, last_requested_object, successes-failures value), where success-failures value updated by walkers; this algorithm produces very small overhead [18]
- GIA: each node sends request to subset of its neighbors, based on it's announced capacity; this algorithm provides ability to make rather bandwidth-efficient searches
- Routing indices (RI): nodes builds thematic indices for stored documents (only documents, not files), and stores some type of "goodness value" for neighbors; this algorithm is bandwidth-effective for

searches, but not for indices and "goodness values" building, updating and calculating [2, 10]

- Distributed Resource Location Protocol (DRLP): if current node has no data matching the query, it forwards query to its neighbors waiting for feedback about number of hits; all subsequent queries are forwarded directly to the neighbors with non zero hits [9]
- Gnutella with shortcuts (GS): original gnutella FS algorithm is extended by making shortcuts to nodes, which are useful to answering queries (returns more results, than others) – so further queries are sending to these nodes [19]; this algorithm is also simple, but it makes network less load-balanced.

## 2 Related work

There are a number of P2P file sharing systems with implemented informed and blind search methods. For example Gnutella/Gnutella2 uses FS method, so their each node's query flood some part of entire network resulting in rather robust and simple, but bandwidth costly approach. SoulSeek uses centralized indices, so it has bad load balancing and vulnerable for technical failures or attacks (like Distributed Denial of Service by generating too many queries by leaf nodes to central indexing node). There are also a number of research systems, developed especially for searching documents: CHORD [13], CAN [11], CRI/ERI/HRI [2], Oceanstore [7], Pastry [12], Tapestry [21]. But the greatest part of them (except *RI) expects for specific network structure and applicable only for document searching.

Search request routing in P2P is also similar to traditional routing algorithms [15]. But while those algorithms are designed to send a packet from one node to another specific one using the shortest way, request routing algorithms in P2P should transmit request packet from one concrete node to non predefined set of other nodes with the goal of returning some number of answers and sources of object being searched.

## 3 Current P2P file sharing leader networks and their features

As it was noticed in the introduction of this article, utilization of internet bandwidth by p2p traffic grows very fast in the past years.

As we can see on the diagram presented below (made by IDG News Service in 11.2007), p2p traffic can utilize up to 70% of total internet bandwidth and main part of this traffic is used by bittorrent/dht [1] and ed2k/kademlia [5].

These networks were designed for effective sharing of huge data amounts (files of different types: video, music, software, etc). So their high popularity may be expected.

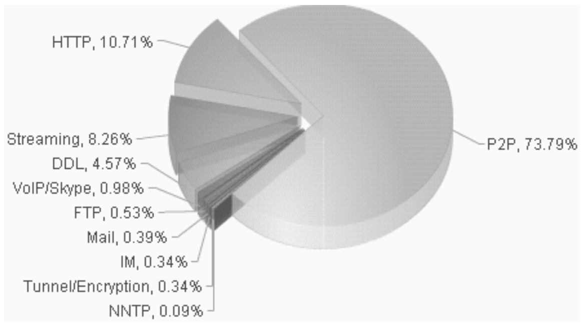

*Fig. 1. Internet bandwidth utilization by different types of traffic – 11.2007, IDG*

Kademlia and bittorrent dht uses a special distributed hash table like structures to store information about shared data and its source nodes. When new node wants to connect to network, it calculates a special hash value that will be used as unique id (node_id) of this node, and than announces it to some other subset (usually random) of known nodes - his immediate neighbors. While process of sharing new files in network, the new node calculates the hashes of these files content, using the same hash function and than announces these file hashes to nodes with the most similar node_id. So when other node wants to send search request to find sources of files with known content hashes, it send this request to most similar node_id neighbors. They forward this query to their neighbors with node_id most similar to hash, specified in query, etc. Usual termination condition is TTL hops or number of sources already found. But this is only file sources search method. In bittorrent dht nodes haven't ability to search files by their name or other criteria. Kademlia allows using file search by their names using previously described ID method. As main terminate condition it uses number of unique file hashes, found during the query processing.

## 4 Bittorrent improvements

As we noticed before, bittorrent and bittorrent dht are the most popular networks nowadays. One of the key advantages of these networks is fast file-transfer protocol and descriptions for every file or group of files in the network. But they also have a big disadvantage: distributed file search by names or descriptions is not suggested and implemented for this network. There was a small Exeem project [3], which allows its users to search files by names in bittorrent using simple FS method for requesting other nodes with Exeem clients, but it wasn't fast, stable and user friendly, so its development has been stopped in 2005.

So we want to suggest some new improvements to bittorrent, based on building thematic indices [10] of descriptions of files. These descriptions are stored on super peer nodes (also called bittorrent trackers) [1]. When some information is being announced by usual leaf node using the tracker, it creates a special file, consists of md5 hashes of each currently announced file in one group. This file should be uploaded to tracker using http protocol with full description of current file group. There are two main tracker engines, most often used as trackers base: tbsource [16] and torrentpier [17], both are free to use and modify. So it's possible to make some modifications to these engines, adding them ability to create thematic indices of uploaded file group's descriptions. Trackers are usually running on high speed hardware, so we can say that load balancing of the network will not be critically decreased. While creating thematic indices we'll use the second main bittorrent feature: all groups of files are united in some big classes, most of them are similar for all trackers. With this feature we can create some set of thematic indexes on every tracker and then using a dht like structure for trackers (not peers) we can share these indices to some neighbor trackers. In this case we expect that every tracker should have multiple id's, according to its different thematic index type hashes. So when client node will send a search request to one tracker, it will be processed, one of the local tracker's thematic indices will be chosen and if number of results and their sources on local tracker is low, this query will be redirected on neighbor trackers with similar local thematic indices. The similarity of thematic index will be chosen by its type, language and main set of index keywords, stored on tracker for all neighbor trackers. These sets of keywords will contain some fixed number of words with maximum hits in each neighbor local thematic index. It should be updated only from time to time – not for every local index update on neighbor trackers. The simple scheme of this idea is shown below.

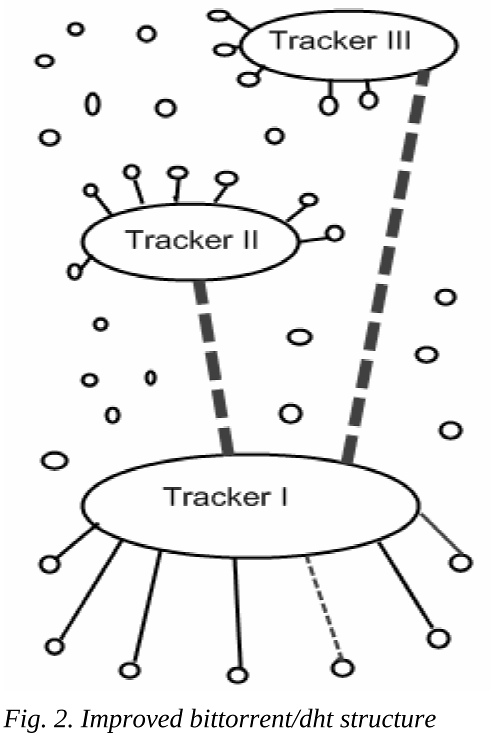

*Fig. 2. Improved bittorrent/dht structure*

Here we can see one peer, sending request (a special type of thin dotted link) to tracker I. Tracker I returns a number of different file group hashes if it can and then forwards this request for some neighbors – for example tracker III, because it have a similar local thematic index and then receives some results from this tracker and a number or sources for this results. At the last stage, Tracker I sends to the initial leaf node merged results, sorted by the number of peers or relevance to the search query. These results include only names of file groups, number of sources and hashes of this file groups. Than initial node can chose one or more of presented results and start download them, finding sources ip addresses and ports by hash in usual way (it can send new request for tracker I for these sources or find them using DHT). The main problem in implementing this feature is special private trackers and a client ratio system on it: every client node reports its uploaded and downloaded amount of data to such tracker and then receives sources for downloading files. This ratio system can prevent clients from downloading huge amounts of data without any uploading to other nodes. When DHT is used in sources searching process, these values can't be checked by tracker for each group of files, so there is a way to fake them. The simple solution for this issue can be made with creating a special hash id for each peer, connected to particular tracker. These ids can be used in reports to private trackers by peers to prevent calculating upload/download statistics for alien nodes. So a node can upload and download data from a large set of nodes and report to tracker all of them, but in their responses, trackers can inform peers, that particular values in statistics are rejected, so these peers can exclude nodes with some ids from further statistic reports to trackers.

But it's clear enough that only subset of tracker's owners will accept and introduce new specification updates listed above. So another way of improving search routing methods in bittorrent/bittorrent dht network is modifying functionality only of usual nodes, not trackers. We took source code of bittornado torrent client and now applying the following modifications to it.

First way of client routing features improving is to give it ability to make its local thematic indices of downloaded files descriptions. These descriptions can be simply grubbed from tbsource or torrentpier based trackers (may be with the help of client's user, providing his trackers login/password, needed to access descriptions via web interface). This improvement gives bittorrent nodes an ability to search files in dht with standard methods (FS/BFS, ID, etc) and also can become a base for further improvement of entire bittorrent dht with adding it functionality to store hashes of local thematic index keywords in the same way, as file content hashes are stored in it or in kademlia network. This will allow applying search queries via bittorrent dht in a native sources search way, used before and using now. The one disadvantage, which can decrease effectiveness of using this method, is different keywords frequency. It can critically increase the number of requests to particular nodes, which have “too popular” hashes as their ids. This problem is not solved yet, but we have some ideas about it, which need some experiments to be done in real bittorrent network segment and with a set of real trackers and clients.

### 4.1. Plan of experiments.

We will use our own bittorrent tracker with the current number or different nodes of about 8600 and some of the other trackers with similar material shared and some equal nodes between them that use the bittornado bittorrent client or original open source bittorrent or mainline clients. Our main goal is to test bittorrent dht network improvements with a set of criterias, which including

- query search time for data, which exists in the network
- query search time for data, which doesn’t exists in the network at the moment, where search request is performed
- number of peers and super-peers involved in random search query processing
- number of unique results returned on successful queries
- number of sources of unique results returned

These parameters will be compared with the same ones for usual bittorrent-dht network, including approximately the same set of nodes (not exactly the same set because of the permanently changing structure of the network) and not using any stored data about nodes content anywhere except one super-node per usual node.

## 5 Conclusion

This paper describes some well known methods of search query routing in p2p file sharing networks like FS/BFS, ID, IW, etc. In this paper we also introduce some methods that can help to improve search request routing methods in widely using p2p file sharing networks, especially bittorrent. No experimental results were given, because our work is not completed yet, but we believe that our approach should be useful for entire bittorrent network and it users and will give it some important features or search queries routing and information sources node discovery.

## References

[1] BitTorrent full specification (version 1.0). http://wiki.theory.org/BitTorrentSpecification

[2] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In ICDCS, July 2002.

[3] Exeem project website. http://www.exeem.it.

[4] Gnutella project official website. http://www.gnutella.com.

[5] Kademlia: A Design Specification. http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html

[6] V. Kalogeraki, D. Gunopulos, D. Zeinalipour-Yazti. A Local Search Mechanism for Peer-to-Peer Networks. In CIKM, 2002.

[7] J. Kubiatowicz, D. Bindel, Y. Chen. Ocean-store: An architecture for global-scale persistent storage. In ASPLOS, 2000.

[8] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In ICS, 2002.

[9] D. Menasce and L. Kanchanapalli. Probabilistic Scalable P2P Resource Location Services. SIGMETRICS Perf. Eval. Review, 2002.

[10] I.Nekrestyanov. Distributed search in topic-oriented document collections. In SCI'99, volume 4, pages 377-383, Orlando, Florida, USA, August 1999.

[11] S. Ratnasamy, P. Francis, M. Handley. A scalable content-addressable network. In ACM SIGCOMM, August 2001.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Middleware, 2001.

[13] I. Stoica, R. Morris, D. Karger. Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. ACM SIGCOMM, 2001.

[14] M. Stokes. Gnutella2 Specifications Part One. http://gnutella2.com/gnutella2_search.htm.

[15] A. S. Tanenbaum. Computer Networks. Prentice Hall, 1996.

[16] TBSource website. http://www.tb-source.info

[17] TorrentPier website. http://torrentpier.info

[18] D. Tsoumakos and N. Roussopoulos. Adaptive Probabilistic Search for Peer-to-Peer Networks. In 3rd IEEE Int-l Conference on P2P Computing, 2003.

[19] D. Tsoumakos, N. Roussopoulos. Analysis and comparison of P2P search methods. Proceedings of the 1st international conference on Scalable information systems, Hong Kong, 2006

[20] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In ICDCS, 2002.

[21] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location androuting. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001