International Journal of Engineering Research ISSN:2319-6890)(online),2347-5013(print)
Volume No.5, Issue No.3, pp : 203-206 1 March. 2016

Comparison of File Sharing Search Algorithms over Peer-to-Peer Networks

Nidal AbidAl-Hamid Al-Dmour
Department of Informaion Technology, Ajman University of Science and Technology Ajman, PO (346),UAE
Corresponding Email : (n.aldmour@ajman.ac.ae)

Abstract: In a Peer-to-Peer system, each participating machine (referred to as a peer) functions as a client with a layer of server functionality; P2P applications build on this functionality for storage, computation, messaging, security, and file distribution. This paper provides a comparison between search algorithms over Peer-to-Peer networks.

Keywords—Peer-to-Peer, Centralized System, Decentralized System, Hybrid Systems, and Search Algorithms..

I. Introduction

In a P2P system, each participating machine (referred to as a peer) functions as a client with a layer of server functionality; P2P applications build on this functionality for storage, computation, messaging, security, and file distribution. A peer can initiate requests and can respond to requests in the system.

P2P networks can be classified according to their topologies. Detecting the presence of other peers is not an isolated activity, and the methodology employed depends upon the topology of the P2P system. In the following section, we describe three kinds of system.

Decentralised Systems

In a decentralised system, there is no central server and peers run independently. Peers handle all communications, and all peers have equal capabilities; each peer operates as both a client and a server. Peers maintain connections to other peers at all times. These connected peers handle the network traffic and reply to the queries from peers searching for resources. These connected peers are also responsible for handling the control messages that facilitate the discovery of other peers.

Any peer can join a network and start exchanging data with another node. A decentralised system can be fault tolerant, as the failure of any particular peer does not impact upon the rest of the systems. Unfortunately, decentralised systems also suffer major drawbacks:

- The search for a resource may require a long time and may need to travel through thousands of peers before the resource is found.
- There is a lack of management, and it is difficult to track the resources that are distributed among peers.
- Decentralised systems are vulnerable to malicious attacks by rogue peers, which may insert frivolous data that reduces the accuracy of searches within it.

Centralized Systems

In a centralized system, the control information is exchanged through a central server (although peers may later act on information received from the central server to contact one another directly). Early P2P applications (e.g. Napster [1]) are centrally coordinated systems. Finding the resources in centrally controlled systems can be done in two ways:

- A peer sends the request to the server, which forwards it to all the peers. The server gathers all the replies and forwards them to the request initiator.
- All peers upload the server with their IDs for their resources (as happened in Napster). The central server can then cross reference peers' requests with its directory database and return any matches. Once the peer is informed about a match, it can directly contact the peer that has the resource.

The main advantage of this topology is the ease of management. The central server acts as a monitoring agent for all the other peers and ensures information coherence. The drawback of a centralized system is a single point of failure: if the server dies, all clients' applications connected to the server also die. A single server also suffers from scalability problems because all control information flows through a single node. Using a better server, or even a cluster, only alleviates the problem and does not solve it completely.

Hybrid Systems

As noted above, decentralised and centralized systems suffer from many drawbacks. Recently, a new variation of both systems has gained popularity for P2P applications. Hybrid systems are centralized systems embedded in a decentralised infrastructure. In this model, certain peers in the network are elected as super-peers. The super-peers change dynamically as the bandwidth and network topology alters: a super-peer may become a normal peer if the network characteristics change. Each super-peer works as a controller for the peers connected to it, and is responsible for the configuration, administration, and security of its peers. Each super-peer maintains two types of communication, super-peer/peer communication and super-peer/super-peer communication. Every peer keeps track of a number of connections between itself and other peers, and must have a connection with a super-peer.

In a hybrid system, the user requests a service, and a peer forwards this request to its super-peer, which then forwards it to other super-peers until the requested resource is found. Super-peers maintain lists of the resources available on the peers that they manage. The advantages of hybrid systems are:

- The number of peers required for message handling and routing is reduced as is the volume of traffic between them.

- The queries for resources from users are answered by the super-peers at a speed comparable with a centralized system. For example, a search which takes *O(N)* time on a centralized/decentralised system, will take *O(N/M)* time on a hybrid system (where *N* is the total number of peers in the network and *M* is the average number of peers connected to a single super-peer).
- Ease of manageability. Super-peers can monitor the peers connected to them, which helps to prevent malicious attacks from rogue users.
- A single point of failure is overcome via the notion of super-peer redundancy, in which a redundant super-peer automatically takes over the job of a failing super-peer.

Table 1 shows that the hybrid system is a more suitable framework for developing P2P applications, because it combines the advantages of both centralized and decentralised systems.

Table.1: *A Comparison of P2P topologies.*

Topology	**Scalability**	**Reliability**	**Manageability**
Centralized	No	No	Yes
Decentralised	Yes	Yes	No
Hybrid	Yes	Yes	Yes

II. File Sharing

File sharing using P2P systems was first brought to public attention through the efforts of Napster. File sharing is the dominant P2P application on the Internet today, allowing users to contribute, search for, and obtain content with greater ease. A P2P system allows users to share files across the network without storing them on a central server. Prior to the rise and fall of Napster, the sharing of files and information between computers was largely confined to the Web, Local Area Networks (LAN) and the exchange of files via the File Transfer Protocol (FTP). The public demand for file sharing technology rose as the speed of personal computers (PCs) and Internet connections increased. The evolution from Napster [1] to KaZaa [2] has dramatically increased the number of users who share their files across the network. Many PCs are now being used as peers that share their files with other peers on the network.

In principle, search engines such as Yahoo and Google can locate these files (if the user creates a Web site and loads it with the files that he/she wants to share). Therefore, other users could locate these files by putting keywords into a search engine. Nevertheless, using Web search technology as an approach for sharing files between users is inefficient for the following reasons:

- Not all the users are capable of creating a Web site and maintaining it. However, P2P applications do not require any administrative efforts from users after a minimal initial set up and configuration.
- Users can share various kinds of files, such as images, and MP3 files. The text search that underlines the search engines is not specialized for these formats. The user who is looking for a certain document using the search engines may receive results that are broader than expected.

There are two types of file sharing system: unstructured and structured. In the next two sections we review the state-of-the-art of each type.

Modified BFS and Intelligent BFS

In Modified BFS, the query is sent to a ratio of neighbouring nodes instead of contacting all of them [3]. Intelligent BFS is an adaption of the Modified BFS [3] algorithm. Nodes store query-neighbour tuples which map classes of queries to neighbours which have answered most of the queries of the related types. When a new query arrives, a peer compares it with the stored queries to find a matching query which has been previously answered. It then chooses to forward the query to a set number of its neighbours that have returned the most results for this query by using a ranking mechanism.

Random Walks

This algorithm [4] aims to reduce the number of messages produced by forwarding a query message to a randomly chosen neighbour at each step until the object is found. This approach cuts down the message overhead significantly by an order of magnitude compared to the previous algorithms [4]. To decrease the delay of receiving successful matches, the requesting node sends *K* query messages instead of one, and each query message takes a Random Walk. The intermediate nodes forward the query messages until the TTL expires or the requested object is found. The most important advantage of Random Walks is the significant message reduction it achieves, since each query produces $k \times TTL$ messages. Results in [104] show that messages are reduced by more than an order of magnitude compared to the original Gnutella.

Gnutella2

In Gnutella2 [5], the nodes are partitioned into ultra-peers and leaf nodes. A node sends its query to the ultra-peers which in turn contacts its neighbouring ultra-peers. These ultra-peers in turn query their leaf nodes. Gnutella2 aims to reduce the effects of flooding by utilizing the structure of hybrid networks. The degree of connectivity for ultra-peers is large – about ten times the degree of connectivity in Gnutella in order to reduce the number of messages and achieve a greater number of successful matches [5].

Local Indices

In this algorithm [156] each node maintains an index of data of all nodes within a certain radius *r*. When a node receives a query, it can process it on behalf of itself or for every node within its radius. A search is performed in a BFS-like manner. To minimize the overhead, the hop-distance between two consecutive depths must be $2r+1$.

GIA

The design of GIA (Gia is short for gianduia, which is the generic name for the hazelnut spread, Nutella) [7] takes into

account the capacity constraints with each node in P2P networks. The search in GIA is based on biased random walks that direct queries toward high-capacity nodes. The capacity of a node is based on a number of factors such as processing power, network latencies, and bandwidth. The algorithm also uses a dynamic topology adaptation protocol that reconfigures the overlay connectivity and puts most nodes within a short reach of high capacity nodes. Those nodes with high capacity receive a large proportion of the queries. Each GIA node actively maintains an index of the content of each of its neighbors. These indices are exchanged when neighbours establish connections. Thus, the biased workers will be directed to the highly connected neighbours to those with the highest number of indexed objects. Finally, GIA uses an active flow control scheme in which a sender is allowed to direct queries to a neighbor only if the neighbour has notified the sender that it is willing to accept queries from the sender.

Routing Indices (RI)

A RI [8] is a data structure used for selecting the best neighbour to send a query to. Goodness of node depends on number of related documents in node and nearby nodes. Documents in this system are classified in zero or more topics, and queries request documents on particular topics (e.g. Algorithms, Networks, Database etc.). Every node contains a local index for local documents and RI. The query termination condition always relates to a minimum number of hits. A node that cannot satisfy the query stop condition with its local repository will forward it to the neighbor with the highest "goodness" value. The number of documents that may be found in a path is used as a measure of goodness of each node for a query. The number of documents is calculated by using the estimator in [9 and 10].

Distributed Resource Location Protocol (DRLP)

Each peer has a Local Directory (LD) that points to the resources managed locally [11]. The contents of the LD cannot be modified or accessed by other nodes. Each peer also has a Directory Cache (DC) that points to the location of the resources managed by other nodes. An entry in DC is a pair (id, loc) where "id" is the object id and "loc" is the location of the object. Initially, when a query message is received, a peer searches its local directory first and then its Directory Cache. If it has no information about the requested object, the peer forwards the query to each of its neighbours with a certain probability. On a hit, the query-node address pair is stored locally in Director Caches of all the nodes in the path. If the same query is encountered again that node is contacted directly. Unfortunately this approach is only suitable for stable networks and cannot cope with node departures or file deletion.

Adaptive Probabilistic Search

This is a modification of Random Walks [12]. Each node maintains a table of entries of the objects it has requested. The value *p* denotes the probability of selecting a node's neighbour in a random selection of finding the next node to forward the query to. Initially, each node sends *K* walkers for each query and if a hit occurs, then it increases the value *p* for each node that forwarded the query; otherwise it is decreased. The disadvantage of this algorithm is that thousands of requests need to be sent by each individual node before it can achieve good results [12].

III. Discussions and Conclusions

In this section we summarize the drawbacks that some of the previous algorithms have.

- Routing Indices: The search with Routing Indices is very bandwidth-efficient. Routing Indices requires flooding in order for the routing tables to be created and updated, so the method is not suitable for dynamic networks. Moreover, stored indices can be inaccurate due to thematic correlations, over- or under-counts in document partitioning and network cycles.
- Napster: The main disadvantages of Napster is its single point of failure as the failure of the central server will stop the peers from sharing the files with each other. Napster is very vulnerable to malicious attacks, and is not scalable. However, the search results from a decentralized approach are less costly as there is no need to have centralized servers and the search results are always fresh.
- Random Walks: The disadvantage of this algorithm is that the successful matches and the number of hits decrease greatly for large networks [4]. Also, the message is randomly forwarded to one of its neighbor without considering whether this node might have the requested object or not.
- Gnutella2: The number of messages has decreased with Gnutella2 as the queries are forwarded to the ultra-peers [5]. However, the search is localized and nodes which are not connected to ultra-peers will never be searched.
- Local Indices: Maintaining an index at a node of all the objects stored at all nodes within a certain radius has increased the number of hits that can be achieved. However, a P2P network is dynamic and peers join and leave at anytime. The dynamicity of P2P networks causes an increase in the number of messages that update the indexes at each node. The number of messages generated with Routing Indices becomes higher than the number of messages generated by Gnutella as the message needs to travel for *r* hops (TTL=*r* for a node that maintains an index of data of all nodes within a radius *r*).
- DRLP and Intelligent BFS: More messages will be generated in a dynamic environment with both algorithms. The performance these algorithms will be fully analysed in the next chapter when a comparison is made between them and our algorithm.
- GIA: Deploying various walkers in the network to discover the available objects causes an increase in the communication overhead in addition to an increase in the responsibility of peers to keep the indexes of objects which they maintain updated as the nodes join and leave the network at any time.

REFERENCES

ISSN:2319-6890)(online),2347-5013(print)
1 March. 2016

i. *Napster project, http://napster.en.softonic.com/, December 2004.*

ii. *KaZaa project, http://www.kazaa.com, February 2016.*

iii. *V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A Local Search Mechanism for Peer-to-Peer Networks", Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM), McLean, Virginia, USA, ACM Press, Pages: 300-307, November 2002, ISBN:1-58113-492-4.*

iv. *C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks", Proceedings of the Sixteenth Annual ACM International Conference on Supercomputing (ICS), New York, USA, ACM Press, Pages: 84-95, June 2002, ISBN: 1-58113-483-5.*

v. *Gnutella2project http://g2.doxu.org/index.php/Main_Page, February 2016.*

vi. *B. Yang and H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks", 22nd International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, IEEE Computer Society, Page: 5, 2002, ISBN:0-7695-1585-1.*

vii. *Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable", Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Karlsruhe, Germany, ACM Press, Pages: 407 - 418, 2003, ISBN:1-58113-735-4.*

viii. *A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems", Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, IEEE Computer Society, Page: 23, 2002, ISBN: 0-7695-1585-1.*

ix. *W. Gentzsch, "Metacomputing from Workstations Clusters to Internet Computing", Future Generation Computer Systems, Special issue on Metacomputing, Volume: 15, Issue: 5-6, Pages: 537-538, 1999, ISSN: 0167-739X.*

x. *C. Germain, V. Neri, G. Fedak and F. Cappello, "XtremeWeb: Building an Experimental Platform for Global Computing", Proceedings of the First IEEE/ACM International Workshop on Grid Computing, Lecture Notes In Computer Science, Volume: 1971, Pages: 91 – 101, 2000, ISBN: 3-540-41403-7.*

xi. *D. Menascé and G. Mason, "Probabilistic Scalable P2P Resource Location Services", ACM SIGMETRICS Performance Evaluation Review archive, ACM Press, Volume: 30, Issue: 2, Pages: 48-58, September 2002, ISSN:0163-5999.*

xii. *D. Tsoumakos and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", The Third IEEE International Conference on P2P Computing, Linkoping, Sweden, 2003.*