An Efficient Search to Improve Neighbour Selection Mechanism in P2P Network

C.R. Totekar and P. Santhi Thilagam

Department of Computer Engineering,
National Institute of Technology Karnataka - Surathkal,
India - 575025
{chinmaytoekar@gmail.com, santhi_soci@yahoo.co.in}

Abstract. One of the key challenging aspects of peer-to-peer systems has been efficient search for objects. For this, we need to minimize the number of nodes that have to be searched, by using minimum number of messages during the search process. This can be done by selectively sending requests to nodes having higher probability of a hit for queried object. In this paper, we present an enhanced selective walk searching algorithm along with low cost replication schemes. Our algorithm is based on the fact that most users in peer-to-peer network share various types of data in different proportions. This knowledge of amount of different kinds of data shared by each node is used to selectively forward the query to a node having higher hit-ratio for the data of requested type, based on history of recently succeeded queries. Replication scheme replicates frequently accessed data objects on the nodes which get high number of similar queries or closer to the peers from where most of the queries are being issued. Two simple replication schemes have been discussed and their performances are compared. Experimental results prove that our searching algorithm performs better than the selective walk searching algorithm.

Keywords: Peer-Peer systems, selective walk, query table, replication schemes.

1 Introduction

Peer-to-Peer systems are increasingly being used nowadays with increasing speed of communication links and high computational power of nodes. Decentralized P2P systems have gained enormous popularity particularly Gnutella [11], Freenet [12] etc. Compared to Centralized P2P systems, which have high probability of single or multiple point of failure. On the other hand purely decentralized system achieves higher fault tolerance by dynamically re-configuring the network as the nodes join and leave.

Structured P2P systems such as CAN [14], Pastry [13], and Chord [15] guarantee to find existing data and provide bounded data lookup efficiency. However, it suffers from high overhead to handle node churn, which is a frequent occurrence of node joining/leaving. Unstructured P2P systems such as gnutella[11] are more flexible in that there is no need to maintain special network structure, and they can easily support complex queries like keyword/full text search. The drawback is that their routing efficiency is low because a large number of peer nodes have to be visited during the search process. The flooding mechanism used by gnutella[11] like systems, results in increased amount of traffic as exponential number of redundant messages are generated during the search process, as the value of TTL is increased.

To address the problems of the original flooding several alternative schemes have been proposed. These algorithms include iterative deepening[4], directed BFS[4], local indices based search[4], random walk[7], Probabilistic search[9], popularity-biased random walk [6], adaptive probabilistic search[5], and dynamic index allocation scheme[2], Selective walk searching[8]. All these techniques try to reduce the number of redundant messages generated because of flooding during search and make search more efficient. These methods either try to maintain the indices of neighboring nodes by each node, or by selecting only those nodes having high probability of having hits, based on recent past.

Selective search scheme uses hints from neighboring peers such as number of shared files, most recent succeeded queries etc to select the most promising candidate for forwarding the queries. However it does not consider the type of data objects that are shared by each node in the algorithm. Most users share files which they themselves are interested in, some might exclusively share movies and some may share only music, and may not share even a single file of picture or document, or archive type. Thus forwarding the query for music file type, to a neighbor having higher share of data as done in Selective Walk searching, might not give proper results if the selected neighbor is sharing no files of type audio, but is having higher share of overall data. Similarly each peer in Selective Walk technique gets rating based on information about most recent succeeded queries maintained in its query table. We can extend this by having a query counter at each node which stores the most recent succeeded queries of each type, thus node having higher value of counter for requested file type gets higher chance of getting selected.

There are very few replication schemes in literature, such as owner replication as used by gnutella and path replication as used by freenet. Gnutella creates copy on the requester node whereas path replication replicates on the path between requester and provider. The third replication algorithm is random replication which is harder to implement. There is need for a simpler approach to find replication site and how many copies to be created.

In this paper, we propose an enhanced selective search which rates each peer based on information about the type of data files it provided recently and number of hits it had. It also uses other details of most recent successful queries at each peer such as list of frequently accessed files and details of total number of different types of files shared by each of the neighboring peers, while selecting the neighbor.

We would like to draw your attention to the fact that it is not possible to modify a paper in any way, once it has been published. This applies to both the printed book and the online version of the publication. Every detail, including the order of the names of the authors, should be checked before the paper is sent to the Volume Editors.

2 Related Work

Several attempts have been made to reduce the number of redundant messages produced during the search which uses flooding. Broadly they can be classified in to two categories: breadth first search and depth first search. Directed BFS [4], iterative deepening [4] improve upon Breadth first search but still have the disadvantage that the number of messages will increase exponentially with increasing TTL. Whereas DFS Based methods such as Adaptive Probabilistic Search [9], Random Walks [7], Selective Walk Searching[8] algorithms perform better as they produce linearly increasing messages with respect to increasing TTL.

Selective walk searching algorithm emphasizes on peer selection criteria, which is based on several hints such as number of files shared by each peer, information about most recent succeeded queries and specific peer that provided the answer. The querying node selects a set of neighbors to forward the query to according to the selection criteria based on these hints. Each of those intermediate peers then selects one peer each to propagate the query until the TTL becomes zero. Each node maintains the details of files shared by its neighbors; neighbor with higher number of files is considered having higher probability of having hit to a query. But this does not consider the type of data file that is queries for, hence there might be situation where one neighbor might have higher number of audio files and another neighbor having higher number of video files, then the neighbor to be selected should be decided based on the type of file requested. If it is request for audio file then the neighbor having highest number of files of type audio should be selected.

Secondly Selective Walk searching algorithm uses a query table at each node to store the most recent succeeded queries, so that if similar query comes at a later stage it will be forwarded according to the past experience. But only if similar query exists then the node will be selected else node with higher shared data will be selected. We use counters at each peer which indicate the number successful queries of each type in the recent past, based on that the neighbor having higher count of recent hits for particular type can be selected.

3 Problem Description

There are a lot of existing techniques for improving search by minimizing the number of redundant messages by selecting most promising neighbor at each step. But no algorithm takes into consideration the type of data shared by each node so that peers can be rated based on this information. Our goal is to design a search scheme for Unstructured P2P system which will decrease the search time, minimize the number of redundant messages, use hints from past to improve selection of neighbor, minimize the number of duplicate messages at each node, and bring the frequently accessed objects closer to the peers from where most queries originate.

4 Proposed Schemes

Here is brief description of the proposed searching technique and the two replication schemes:

4.1 Improved Selective Walk Searching

In selective walk search method type of data file requested is not considered hence no details about the file type is maintained at the query tables, and details about the different types of data shared by the neighbors. Improved selective walk searching scheme considers these details to provide better hints and an improved selection criteria to use these hints.

Information Maintained by nodes: Each node maintains three types of information: most recent succeeded queries in Query Table, a counter for each data type which stores the count of recent successful queries for each type of data, and total number of files shared of each type. Each of these information units will be exchanged with neighbors periodically, so that each node will have these details of each of their neighbors. The Query Table maintains the details of most recent successful queries such as query id, number of hits, Search criteria and the file type. Each node maintains a counter for each type of data which keeps the count of recent hits for each type of data and total number of files of each type that are shared. The structure of the query table is given in Fig. 1.

Query Id	Number of Hits	Search Criteria	Type

Fig. 1. Format of Query Table

Selection of neighbor: Selection of neighbor is done based on hints such as most recent successful queries listed in the query tables of neighbors, Counter for successful queries for each type, and total number of files shared of each type. The searching node will select a fraction of the neighbors to forward the query to. Suppose the total number of neighbors is N then n neighbors will be selected based on following criteria:

(i) First all the nodes that have similar object in their Query tables will be selected in order of the number of hits, until number of selected neighbors does not exceed n.

(ii) If the number of selected neighbors is still less than ‘n’ than select the neighbors having higher number of shared files of requested file type until their number reaches n.

Once the query is passed to neighboring nodes they will intern forward the query to exactly one node by using following criteria: Look up the query tables of neighbors, if there is an entry already existing in neighbors query table then select the neighbor having highest number of hits for that object, else select the neighbor having highest number of recent hits for the requested file-type. The queries will be forwarded until the TTL becomes zero. This is depicted in Fig. 2.

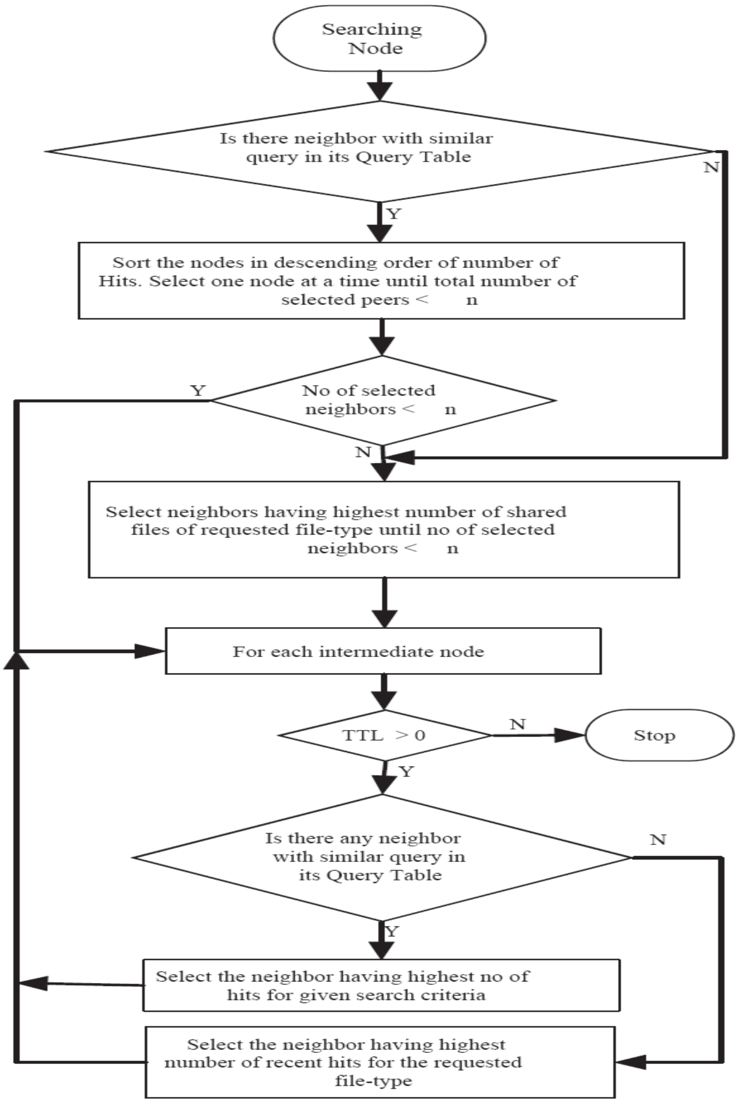
Searching Node
Is there neighbor with similar query in its Query Table
N
Y
Sort the nodes in descending order of number of Hits. Select one node at a time until total number of selected peers < n
Y
No of selected neighbors < n
N
Select neighbors having highest number of shared files of requested file-type until no of selected neighbors < n
For each intermediate node
TTL > 0
N
Stop
Y
Is there any neighbor with similar query in its Query Table
N
Y
Select the neighbor having highest no of hits for given search criteria
Select the neighbor having highest number of recent hits for the requested file-type

Fig. 2. Criteria for Neighbor Selection

Updating of Query Table: When the query is formed, each query has unique Query-Id, which gets recoded at each of the intermediate nodes. When there is a hit at a particular node that node sends Query-Hit message in response in the reverse path back to the query source, thus each node updates its query table on this path. This is done by having Unique Id for Query-Hit same as that in the Query message. Similarly the Hit counter for that particular data type is to be incremented at each of the nodes.

5 Experimental Study

Assumptions: Some assumptions have been taken during the implementation of the proposed algorithm. Some of them are:

(1) All nodes are having equal bandwidth and processing-power.
(2) The communication delay between each pair of nodes is same and equal to 1.
(3) All the nodes will be up and running throughout the search process.

Experimental Setup: Simulation environment consists of 64 nodes, each containing between 1000-4000 files, each node connected to 1 - 8 nodes, with an average 3 connection per node. The cost of links between peers is fixed and equal to 1. Distance between the farthest nodes is 9. The file sizes are between 1 and 5MB for audio, 5 to 700MB for Video, 10KB to 1MB for text and 5 to 200MB for archives. The size of Query Table is taken as 500, which means 500 recent succeeding Queries are maintained in Query Table. Each of the nodes are having a separate available disk space, of varying capacities, where the frequently accessed data can be replicated. A randomly chosen peer will generate search query for a file using TTL equal to 9. The total cost of searching a file is calculated. A total of 8000 such searches are initiated and total cost of 10000 searches is computed. For determining the performance of the replication schemes initially there exists exactly one copy of file in the entire network. Replication will be happening in background during the search process, whenever any node gets overloaded. The performances are measured at the end of search process.

Performance Metrics: Scalability, efficiency and responsiveness are the main properties of good searching algorithm. An efficient search algorithm should generate minimum number of messages and at the same time should have high hit rate too. Below are some of the performance metrics used in evaluation and comparison of our algorithms.

Query Efficiency: It can be defined as ratio of query hits to messages per node.

Query Efficiency = Query Hits / Msgs Per Node.

Search responsiveness: A second criterion for search performance is search responsiveness, which reflects how quickly the system responds to user requests.

Responsiveness = SuccessRate / No.of.hops.

Therefore *Search Efficiency* can be defined as Query Efficiency x Search Responsiveness.

Average Time for first hit: Average Time for first hit is the time taken by searching node to get the first response.

Number of duplicate messages: Number of duplicate messages reflect the number of query messages dropped by nodes when same query is received more than once.

6 Results and Analysis

The searching algorithm is implemented and compared with Selective Walk search. Simulation was performed with a total of 8000 searches, and performance recorded as shown below in the graphs. Query efficiency and search responsiveness of our search scheme are both better compared to the Selective Walk (SW) which is shown in Fig. 3 and Fig.4. Search efficiency of our algorithm is 0.65 as against 0.4 of Selective walk Searching scheme which is shown in Fig. 5. Our algorithm again results in better success rate, approximately 15% more than the SW scheme. Average time taken by queries is also improved using our scheme which is shown in Fig. 6 and Fig.7.

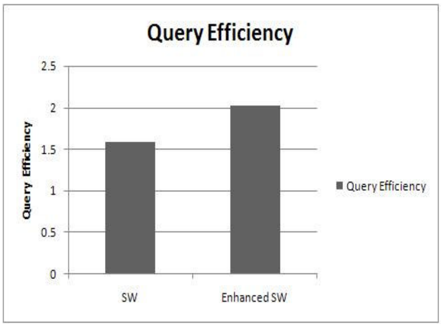

Fig. 3. Query Efficiency

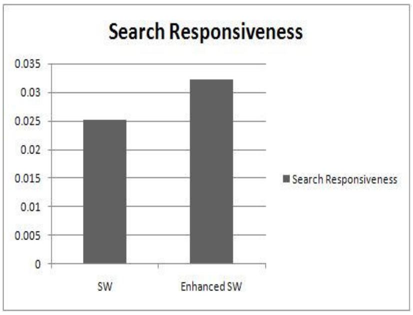

Fig. 4. Search Responsiveness

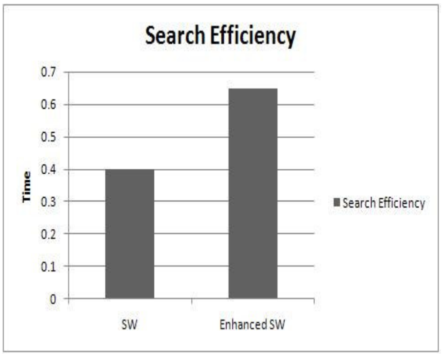

Fig. 5. Search Efficiency

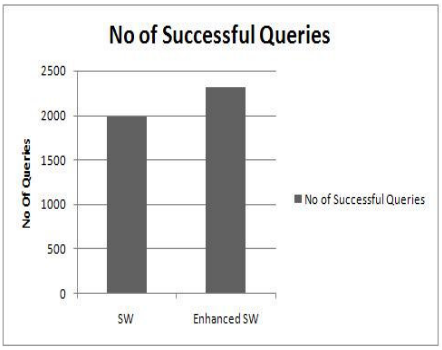

Fig. 6. Number of successful queries

Average Time for first hit is better in our case as shown in Fig. 8. Our algorithm performs better in reducing number of such duplicate messages as shown in Fig. 9 and Fig.10, which in turn will help to reduce loss of messages at node due to overloading, when a node gets too many queries. Search efficiency of our algorithm is 0.65 as against 0.4 of Selective walk Searching scheme which is shown in Fig. 5. Our algorithm again results in better success rate, approximately 15% more than the SW scheme. Average time taken by queries is also improved using our scheme which is shown in Fig. 6 and Fig.7. Average Time for first hit is better in our case as shown in Fig. 8. Our algorithm performs better in reducing number of such duplicate messages as shown in Fig. 9 and Fig. 10, which in turn will help to reduce loss of messages at node due to overloading, when a node gets too many queries.

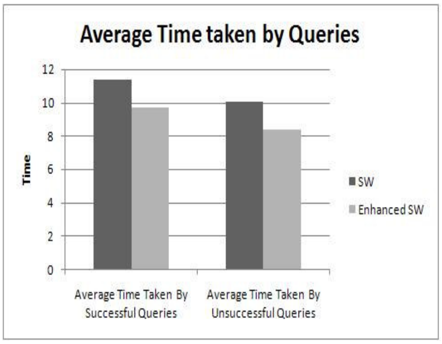

Fig. 7. Average Time for successful queries

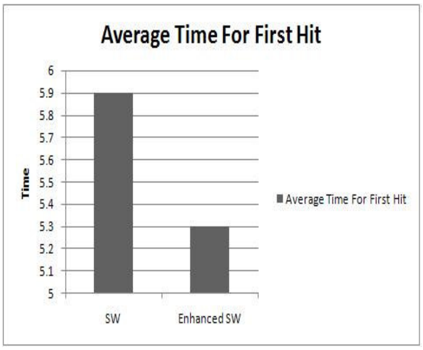

Fig. 8. Average Time for first Hit

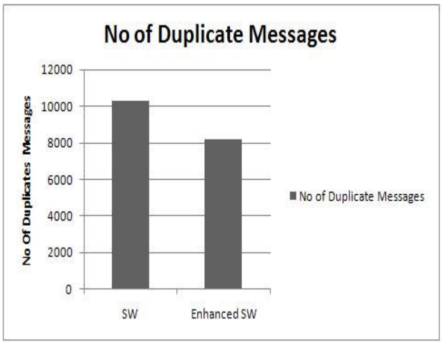

Fig. 9. Number of duplicate messages

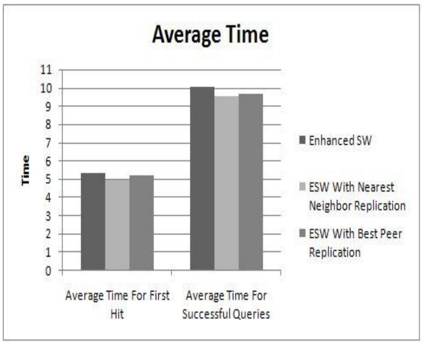

Fig. 10. Comparison of search with the two-replication schemes and without replication

Finally the comparison of the two replication schemes Nearest Neighbor replication and Best Neighbor replication show that Nearest Neighbor replication has slight better performance than Best Neighbor in terms of average time for successful queries and average time for first query. Both the algorithms result in decrease in Average time for first hit and Average time for successful queries.

7 Conclusions

The paper presents the design and evaluation of an efficient search scheme which tries to improve neighbor selection mechanism by rating each peer based on past history and contents shared by node. Our searching scheme performs better, at least as good as Selective Walk scheme in most of the criteria's used for evaluation. It has more success rate, better search response and decreased number of duplicate messages. We have also presented and evaluated two replication schemes which in turn help to reduce the search time by replicating the frequently accessed objects closer to query issuers. Because of the simplicity of these replication schemes and their performance benefits they can be easily incorporated in real world P2P networks.

References

1. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. In: Proceedings of the 16th International Conference on Supercomputing, pp. 84–95 (2002)
2. Ohta, T., Masuda, Y., Mitsukawa, K., Kakuda, Y., Ito, A.: A Dynamic Index Allocation Scheme for Peer-to-Peer Networks. In: ISADS 2005, pp. 667–672 (2005)
3. Zheng, Q., Lu, X., Zhu, P., Peng, W.: An Efficient Random Walks Based Approach to Reducing File Locating Delay in Unstructured P2P Network. In: Global Telecommunications Conference, vol. 2(5). IEEE, Los Alamitos (2005)
4. Yang, B., Garcia-Molina, H.: Improving Search in Peer-to-Peer Networks. In: Proceedings of the 22*nd* International Conference on Distributed Computing Systems (2002)
5. Tsoumakos, D., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. In: Proceedings of the Third International Conference on Peer-to-Peer Computing (2003)
6. Zhong, M., Shen, K.: Popularity-biased random walks for peer to peer search under the square-root principle. In: Proceedings of the 5*th* International Workshop on Peer-to-Peer Systems, CA (2006)
7. Gkantsidis, C., Mihail, M., Saberi, A.: Random Walks in Peer-to-Peer Networks. In: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1(12) (2004)
8. Xu, Y., XiaoJun, Wang, M.C.: Selective Walk Searching Algorithm for Gnutella Network. In: Consumer Communications and Networking Conference, CCNC, pp. 746–750 (2007)
9. Cheng, A.-H., Joung, Y.-J.: Probabilistic File Indexing and Searching in Unstructured Peer-to-Peer Network. In: CCGrid 2004, pp. 9–18 (2004)
10. Zhao, J., Lu, J.: Overlay Mismatching of Unstructured P2P Networks using Physical Locality Information. In: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, pp. 75–76 (2006)
11. Official Gnutella website, `http://www.gnutella.com`
12. Official FreeNet website, `http://www.freenetproject.org`
13. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
14. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proceedings of ACM SIGCOMM, pp. 161–172 (2001)
15. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peertopeer Lookup Service for Internet Applications. In: Proceedings of ACM SIGCOMM, CA (2001)