Search Improvement In Unstructured P2P Network Considering Type of Content

Chinmay R Totekar,Vani M, Palavalli Sharath R
Department Of Computer Engg., NIT Karnataka, Surathkal, India.
Email:chinmaytotekar@gmail.com, Email:vani@nitk.ac.in Email: palavalli.sharath@gmail.co

***Abstract*- One of the key challenging aspects of peer-to-peer systems has been efficient search for objects. To achieve this, we need to minimize the number of nodes that have to be searched, thereby use 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 the queried object. In this paper we present an algorithm CBWS, for searching in unstructured peer-to-peer network, which is based on the fact that most users in peer-to-peer network share various types of data(e.g. audio, video, text, archive) in different proportions. The information about the number of objects of each file-type shared by nodes, is used to selectively forward the query to a node having higher hit-ratio for the data objects of requested type, based on the history of recently succeeded queries. Simulation results prove that our searching algorithm performs better than the selective walk searching algorithm.**

***Keywords*- Peer-To-Peer, Search, Unstructured, Performance.**

I. INTRODUCTION

Peer-to-peer systems are increasingly being used now-a-days with increasing speed of communication links and high computational power of nodes. Decentralized P2P systems have gained enormous popularity, particularly gnutella [10], freenet [11] 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[12], Pastry[11], and Chord[13] guarantee to find existing data and provide bounded data lookup efficiency. However, it suffers from high overhead due to frequent node joining and leaving.

Unstructured P2P systems such as gnutella [10] 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 like systems results in increased amount of traffic, as exponential number of redundant messages are generated during the search process with increasing value of *TTL*.

To address the problems of the original flooding, several alternative schemes have been proposed. These algorithms include iterative deepening[3], directed BFS[3], local indices based search[3], random walk[6], probabilistic search[8], popularity-biased random walk[5], adaptive probabilistic search[4], and dynamic index allocation scheme[1], selective walk searching[7].

All these techniques try to reduce the number of redundant messages generated because of flooding during search, thereby making search more efficient. These methods either ask each node to maintain the indices of its neighboring nodes, or select only those nodes with high probability of returning a hit, based on information about the recently succeeded queries.

Selective walk searching[7] 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, some might exclusively share movies, while some might share only music. 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 the information about most recent succeeded queries maintained in its Query Table. We can extend this by having Query Counters at each node which store the count of most recent succeeded queries of each type, thus node having higher value of counter for requested file type gets higher chance of getting selected.

In this paper we propose a search scheme which rates each peer based on information about the files it provided recently and the number of hits it had of each type. It uses these statistics and details about total number of different types of files shared by each of the nodes, to select the neighbor while forwarding the query.

Our paper is organized as follows. First our search scheme is discussed, later the simulation environment and performance metrics are described and finally the results are analyzed.

II. PREVIOUS 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 [3],

Iterative Deepening[3] 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 [8], Random Walks [6], Selective Walk Searching[7] 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 the 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 queried 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, than the neighbor to be selected should be decided based on the file-type requested. If it is request for audio file, than 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 selection criteria. But only if similar query exists, than 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 which the neighbor having higher count of recent hits for particular type can be selected.

III. PROPOSED SCHEME

Here is brief description of the proposed searching technique, Content Based Selective Walk (CBSW).

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. Content Based Selective Walk searching scheme considers these details to provide better hints and an improved selection criteria which makes use of these hints.

A. Information maintained by nodes

Query- Id	Number Of Hits	Search Criteria	Type

Fig. 1. Format of Query Table

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 the total number of files shared of each file-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 too. The Query Table maintains the details of most recent successful queries such as Query-Id, number of hits, search criteria and the file type.

B. 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 of each type, and total number of files of each type shared by the nodes.

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:

1) First all the nodes that have similar object in their Query tables will be selected, with node having highest number of hits being selected first, until number of selected neighbors does not exceed *n*.

2) If the number of selected neighbors is still less than *n,* then select the neighbors having higher value of μx(total number of shared files of requested file-type) + λ x (total number of hits for requested file-type), until *n* peers are selected. λ and μ being constants.

Once the query is passed to the neighboring nodes with a fixed value of *TTL*, each of them will in-turn forward the query to exactly one node. While forwarding, it looks up the Query Tables of its neighbors maintained by it. If there is an entry already existing in neighbors Query Tables, than it selects the neighbor having highest number of hits for that particular object, else the one having higher value of μ x (total number of shared files of requested file-type) + λ x (total number of hits for requested file-type). The queries will in-turn be forwarded until *TTL* becomes zero.

C. Updating of Query Table

When the query is formed each Query has a unique Query-Id, which gets recorded 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 the unique Query-Id for Query-Hit same as that in the Query message. Similarly the Hit Counter for that particular file-type is incremented at each of the nodes.

IV. SIMULATION

A. Experimental Setup

Simulation environment consists of 128 nodes, each containing between 1000-4000 files, each node connected to 1-15 nodes, with an average 6 connections per node. The cost of links between neighboring peers is fixed and equal to 1. Distance between the farthest nodes is 10 hops. 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 most recent succeeding queries are maintained in the Query Table.

A randomly chosen peer will generate search query for a file using TTL equal to 10. The total cost of searching a file is

calculated. A total of 8000 such searches are initiated and total cost of 8000 searches is computed.

B. 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: Query Efficiency 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 is defined as Success Rate/Noofhops. Search Efficiency is product of Query Efficiency and 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 reflects the number of query messages dropped by nodes when same query is received more than once.

V. RESULTS AND ANALYSIS

The performance of Content Based Selective Walk (CBSW) is compared with selective walk (SW) 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). Search efficiency of our algorithm is 0.66 as against 0.41 of Selective Walk Searching scheme.

Our algorithm again results in improved success rate; approximately 15%more than the SW scheme. Average time taken by queries is also improved using our scheme.

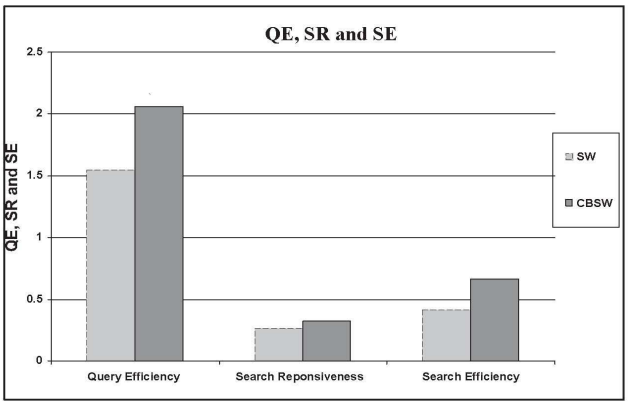

Fig. 2. Search Efficiency

Average Time for first hit is better in our case. Our algorithm performs better in reducing number of duplicate messages, which in-turn will help to reduce dropping of messages at nodes due to overloading, when a node gets too many queries.

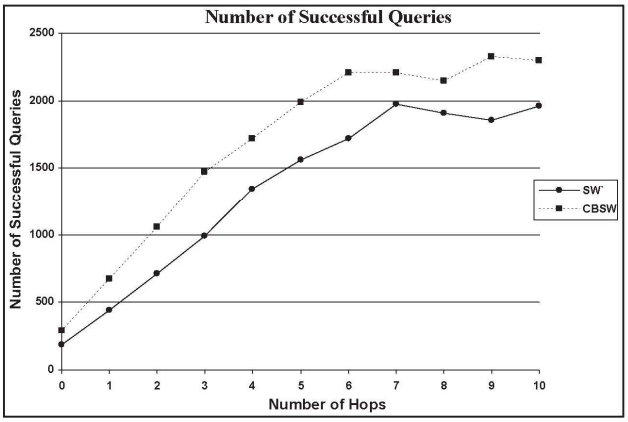

Fig. 3. Number of successful queries

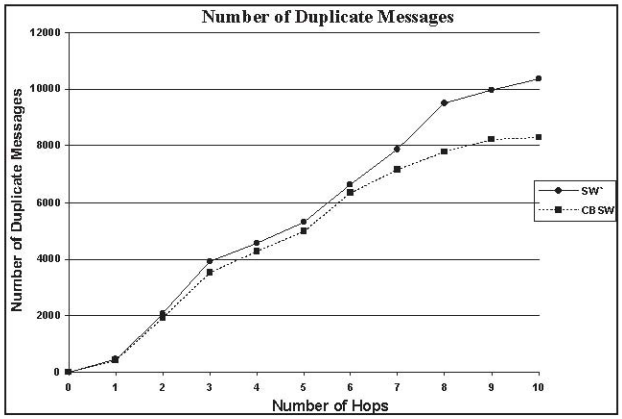

Fig. 4. Number of duplicate messages

Fig. 5. Average Time for first hit

VI. CONCLUSION

This paper presents the design and evaluation of an efficient search scheme with improved neighbor selection mechanism of peers while forwarding queries, by rating each peer based

on past history and contents shared by nodes. Content Based Selective Walk performs better, at least as good as Selective Walk scheme in most of the criteria used for evaluation. It has

higher success rate, better search response and reduced number of duplicate messages. Because of its performance benefit and simplicity CBSW can be easily incorporated into real world P2P networks.

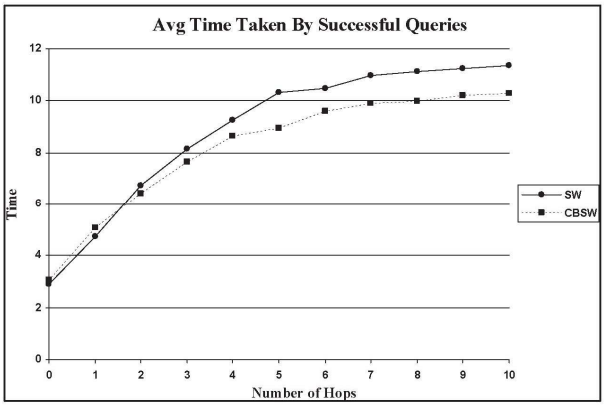

Fig. 6. Average Time for successful queries

REFERENCES

[1] Tomoyuki Ohta, Yasuo Masuda, Kouichi Mitsukawa, Yoshiaki Kakuda, and Atsushi Ito, "A Dynamic Index Allocation Scheme for Peer-to-Peer Networks", Autonomous Decentralized Systems, ISADS 2005, Pages: 667-672.

[2] Qianbing Zheng, Xicheng Lu, Peidong Zhu, Wei Peng,"An Efficient Random Walks Based Approach to Reducing File Locating Delay in Unstructured P2P Network", Global Telecommunications Conference, IEEE Volume 2, Page(s):5 GLOBECOM '05.

[3] Beverly Yang,Hector Garcia-Molina, "Improving Search in Peer-to-Peer Networks", Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), IEEE.

[4] Dimitrios Tsoumakos and Nick Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P03), IEEE.

[5] Ming Zhong, Kai Shen, "Popularity-biased random walks for peer to peer search under the square-root principle", In Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS), Santa Barbara, CA, February 2006.

[6] Christos Gkantsidis, Milena Mihail, and Amin Saberi, "Random Walks in Peer-to-Peer Networks", Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2004. Volume 1, Page(s):12

[7] Yan Xu, XiaoJun,Ma Charles Wang,"Selective Walk Searching Algorithm for Gnutella Network", Consumer Communications and Networking Conference, CCNC ,IEEE 2007, Page(s):746-750.

[8] An-Hsun Cheng, Yuh-Jzer Joung, "Probabilistic File Indexing and Search-ing in Unstructured Peer-to-Peer Network", Cluster Computing and the Grid, CCGrid 2004, IEEE, Page(s):9-18

[9] Official Gnutella website, www.gnutella.com.

[10] Official FreeNet website, www.freenetproject.org.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems", IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany. Pages:329-350.

[12] Sylvia Ratnasamy,Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. "A Scalable Content-Addressable Network", Proceedings of ACM SIGCOMM 2001, Pages: 161-172.

[13] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan, "Chord: A Scalable Peer to peer Lookup Service for Internet Applications", Proceedings of ACM SIGCOMM 2001, SanDeigo, CA, August 2001.