International Journal of Soft Computing 4 (6): 243-249, 2009
ISSN: 1816-9503
© Medwell Journals, 2009

Improved Search Efficiency in Unstructured Peer to Peer Networks Using Search Result Path Caching

S. Anbu and K.P. Thooyamani
Department of Computer Science and Engineering, Bharath University, Chennai, India

Abstract: The huge popularity of recent Peer-to-Peer (P2P) file sharing systems has been mainly driven by the scalability of their architectures and the flexibility of their search facilities. Such systems are usually designed as Unstructured P2P networks. So, designing an efficient search algorithm is a key challenge in unstructured peer-to-peer networks due to the unstructured paradigm. In this study, we proposed a Search Result Path Caching algorithm (SRPC). The proposed scheme combines the Dynamic Search (DS) algorithm and result path caching for effective search. This proposed algorithm takes the advantages from dynamic search and path caching technique works with the queried results. After the successful discovery of results the queries are returned to originator, according to our proposed algorithm the originator will stores the results for future references. In future stored references are used to search the information without querying the overall network. We analyze the performance of our algorithm based on some performance metrics including the success rate, search time and search efficiency. The numerical results shows that the proposed SRPC algorithm performs about 5 times better than DS, 125 times better than flooding and 275 times better than Random Walk (RW) in power-law graphs.

Key words: Peer-to-Peer system, dynamic search, search efficiency, query flooding, path catching, filtering mechanism, new man's random graph

INTRODUCTION

Peer-to-Peer (P2P) systems have gained tremendous momentum in recent years. These networks can be largely classified into two categories, namely, structured P2P networks based on a Distributed Hash Table (DHT) Rowstron and Druschel (2001) and unstructured P2P networks based on diverse random search strategies (e.g., flooding) (The Gnutella, 2003). Due to the emergence of sophisticated overlay structures, unstructured P2P systems remain highly popular and widely deployed. It exhibits many simple yet attractive features, such as low-cost maintenance and high flexibility in data placement and node neighborhood. Unstructured P2P systems are particularly used in file-sharing communities due to their capacity of handling keyword queries, i.e., queries using some key words instead of the whole filename. Unfortunately, as these systems grow in popularity, they aggressively exploit network resources, typically by consuming huge amounts of bandwidth. This issue leads inevitably to performance deterioration and user's dissatisfaction.

Peer nodes do not have any global information about the whole topology and the location of queried resources in unstructured P2P network. Because of the dynamic property of unstructured P2P networks, capturing the global behavior is also difficult (Rasti *et al*., 2006). So, it is a difficult task to give an efficient searching algorithm. Search algorithms provide the capabilities to locate the queried resources and to route the message to the target node. Due to the data deficit the efficiency of search algorithms is critical to the performance of unstructured P2P networks (Milojicic *et al*., 2003). Previous search algorithms in unstructured P2P networks can be classified into two categories: Breadth First Search (BFS) based methods and Depth First Search (DFS) based methods. These two types of search algorithms tend to be inefficient, because they generates too much load on the system (Sripanidkulchai, 2001) or not meeting the users' requirements. Flooding, which belongs to BFS-based methods, is the default search algorithm for Gnutella network (Kan, 2001). In this method, the query source sends its query messages to all of its neighbors. When a node receives a query message, it first checks if it has the queried resource or not. If yes, it sends a response back to the query source to indicate a query hit. Otherwise, it sends the query messages to all of its neighbors, except for the one the query message comes from. The drawback of flooding is the search cost and it produces considerable query messages even when the resource distribution is scarce. If the network grows unlimited from the query source, the number of query messages generated by flooding at each hop would be based on neighbors. There is a possible chance of endless loop of

Corresponding Author: S. Anbu, Department of Computer Science and Engineering, Bharath University, Chennai, India

query messages due to the incorrect routing. So, the flooding is conducted in a hop-by-hop fashion counted by a Time-To-Live (TTL) count. TTL is a packet that tells a network router whether or not the packet has been in the network too long and should be discarded. A message starts off with its initial TTL, which is decremented by one when it travels across one hop. A message comes to its end either because it becomes redundant message or because its TTL is decremented to 0.

In this study, we proposed the SRPC algorithm, which is a generalized from DS algorithm with path caching technique. Our proposed technique is designed to eliminate the drawbacks of the previous algorithms. From the blind techniques of flooding and random walk, we construct the probabilistic function based on the information learned from the past experiences, with respect to each search target, search time and local topology information. Therefore, a node has more information to intelligently decide how many query messages to send and to which peers these messages should be forwarded. After the query hit the results will be returned and stored by the originator for the future reference. Whenever, the node receives the query message first it searches the stored path results and then it will send the query message to the neighbors. Otherwise, it will send the queried resource path information to the queried one from the stored path results. The proposed algorithm do invalidation of path information stored in the originator, which was done by a predetermined time interval. At the time of invalidation the originator sends the path verification message to the destination source. If it receives the positive return, keeps the location information otherwise removes from the path result hash table. This technique helps to maintain the valid path information in the node itself. It avoids the invalid message query over the network.

MATERIALS AND METHODS

There is a lot previous research done by researchers for unstructured Peer-to-Peer searching. The most of the techniques are developed to improve the efficiency of search, minimize the network traffic and search time, such as Modified BFS (MBFS) Kalogeraki *et al.* (2002) directed BFS (Yang and Garcia-Molina, 2002) expanding ring (Lv *et al*., 2002a) and Random Periodical Flooding (RPF) Zhuang *et al*. (2003). These try to modify the operation of flooding to improve the efficiency.

Tsungnan *et al.* (2009) presented DS algorithm it is a generalized form of flooding and random walk. The operation on DS is based on flooding and random walk for short-term search and long-term search, respectively. This was developed by the past experience of the drawbacks of the both flooding and random walk. The disadvantage of flooding is the searching cost and random walk could not a reliable one, so these are considered for the development of DS. The DS uses the knowledge-based approach to discover a resource. This knowledge-based algorithm takes the advantage of the knowledge from previous search results and route the query message based on that. If a search request is out of a node's knowledge, this node would perform a flooding search.

Gang *et al*. (2008) proposed User Interest Model (UIM) based on a general probabilistic modeling tool termed Condition Random Fields (CRFs) (Lafferty *et al*., 2001). From this model, we able to estimate the probability of any peer sharing a certain resource upon given the fact that it shares another resource. This estimation gives rise to an interest distance between any two peers. A greedy file search protocol is presented in this study for fast resource discovery. Whenever, a node receives a query for a certain file that is not available locally, it will forward the query to one of its neighbors that have the highest probability of actually sharing that file. Here, the neighbors list was maintained by a routing table updating protocol. It was done with the help of a newly introduced filtering mechanism, the whole P2P network will gradually self organize into a small world. Filtering mechanism is introduced to mitigate the impact of uneven updating.

The major problem in the unstructured P2P searching by flooding is the generation of redundant messages and it will down the overall performance and maximize the network traffic. When multiple messages with the same message ID are sent to a peer by its multiple neighbors, all, except for the first message, are considered as redundant messages. These redundant messages are pure overhead: they increase the network transfer and peer processing burden without enlarging the propagation scope. As a result, some researchers believe that fully decentralized Gnutella-style systems with flooding do not scale (Lv *et al*., 2002b; Ritter, 2001). Pure flooding starts with a fixed TTL from a peer to reach its neighborhood within its radius, adaptive flooding takes more dynamic factors into consideration to reduce the flooding range while maintaining the necessary search quality. In the expanding ring (Lv *et al*., 2002c) several successive flooding searches are initiated with increasing TTLs. After each flooding, the requesting peer has to wait for some period of time to collect response messages. To overcome this problem by Jiang *et al.* (2008) proposed an efficient flooding scheme, called LightFlood. It retains the merits of pure flooding and also can eliminate most of the redundant messages caused by pure flooding. This consists of two observations, first one is the majority of redundant messages are generated within high hops and the second one is the network coverage growth rates in low hops are much higher than those within high hops.

Search result path caching algorithm: In this study, we provide the details of the proposed Search Result Path Caching algorithm (SRPC). This study presents the operation of SRPC algorithm and provides the Invalidation of the Path information.

Operation of search result path caching algorithm: SRPC is a generalization of DS. Our proposed algorithm has three phases. Each phase has different searching strategies. The choice of search strategy at each phase depends on the relationship between the hop count h of query messages, queried resource q, the decision threshold n and the hash table for storing search result paths z of SRPC.

Phase 1: When (z has the location information of q), At this phase, SRPC searches the queried source is available in the hash table for search result or not. If it discovers the queried source returns the location information to the query source. Otherwise, it continues the second phase.

Phase 2: When h<n, At this phase, SRPC acts as flooding or MBFS according to Dynamic search nature. The predefined transmission probability p is used to send the query message to the number of neighbors. The query messages are send to only link degree d of p neighbors. If p is equal to 1 SRPC continues flooding otherwise it transmissions probability p with MBFS.

Phase 3: When h>n, At this phase, the search algorithm prefers Random Walk. If it does not have the queried resource the node sends the query message to the one of the neighbor which was selected by randomly. The searching ends after achieving the queried resource or TTL reaches 0. If it discovered the queried resource, sends the location information and stops searching.

Algorithm: The pseudo-code of SRPC

Input: ‘s’ query source, ‘p’ transmission probability, m_i is ith query message, ‘r’ vertex that receive the query message, ‘z’ hash table for SRPC, ‘t’ time period for refreshing path information.

Output: The location information of f

```
SRPC(s, q, p)
/* the operation of s */
h = 0
if (z has the location information of q)
    returns the information to s
    originator stores the path
   information in path hash table
    $m_i$ stops
if (h<= n)
    h = h+1
  s choose p portion of its neighbors
  $m_i$ carring h visits these chosen
  neighbors
elseif (h>n)
    h = h+1
    s choose p portion of its
       neighbors

    $m_i$ carring h visits one neighbor of s
/* the operation of r */foreach (r)

if (z has the location information of q)
    returns the information to s
    originator stores the path
    information in z
    $m_i$ stops
elseif (r has the location information of q)
    r  returns the information to s
    originator stores the path
    information in z
    $m_i$ stops
elseif (h>TTL)
    $m_i$ stops
elseif (h<=n)
    h = h+1
    r choose p portion of its neighbors
    $m_i$ carring h visits these chosen neighbors
elseif (h>n)
    h = h+1
    $m_i$ carring h visits one neighbor of r

/* refreshing Path information */foreach (t)
    foreach (r)
        r verifies all stored paths in z are valid or not
        if (available)
           continue
else
           removes the path
           information from the z
```

Invalidating or removing the path informations: According to the predetermined time interval t each node sends the verification message to all stored or available paths.

If it receives the positive return, keeps the path information otherwise removes or invalidates from the stored path hash table.

Performance evaluations: In this study, we present the performance evaluation of SRPC. We apply Newman *et al*. (2001)’s random graph as the network topology, adopt the generation functions to model the link degree distribution and analyze SRPC based on some performance metrics, including the success rate, search time, query messages, query efficiency and search efficiency.

Due to the difficulties to correctly measure and sample the operational P2P networks, there are only limited real data about the topologies of such networks. In this study, we will use the two top most common topologies, the power-law graphs and the bimodal topologies, to evaluate the search performance.

Network model: First, we summarize Newman *et al.* (2001)'s research about the random graph. Let $G_0(x)$ be the generating function for the distribution of the vertex degree k. $G_0(x)$ can be represented as:

$$G_0(x) = \sum_{k=1}^{m} pkx^k \qquad (1)$$

Where:

pk	=	The probability that a randomly chosen vertex
(x)	=	The graph has degree k and m is the maximum degree

Performance analysis

Success Rate (SR): The success rate is the probability that a query is successful, i.e., there is at least one query hit:

$$SR = 1 - (1 - R)^C \qquad (2)$$

R = The replication ratio and C is the coverage

Search Time (ST): To represent the capability of one search algorithm to find the queried resource in time with a given probability, we define the search time ST as the time it takes to guarantee the query success with success rate requirement SRreq. ST represents the hop count that a search is successful with a probabilistic guarantee.

$$ST_{MBFS} = {}^{\log}p.G'_1(1)(((p.G'_1(1) - 1).\log_{(1-R)} (1 - SRreq)/p.G'_0(1)) + 1) \qquad (3)$$

$$ST_{RW} = (\log_{(1-R)}(1 - SR_{req}) - \sum_{i=0}^{t-1} G'_0(1).(G'_1(1))^i)/k) + t \qquad (4)$$

$$ST_{DS} \approx n + (\log_{(1-R)}(1 - SR_{req})/ p^n.G'_0(1).(G'_1(1)^{n-1}) - 1 \qquad (5)$$

We can define the search time for SRPC from the above equations. If node's path hash table has the path information of requested source then ST of SRPC becomes:

$$ST_{SRPC} \approx \text{path hash table evolution time} \qquad (6)$$

It is very less time, we can also negligee this time. If node's path hash table do not has the path information of requested source then ST of SRPC becomes:

$$ST_{SRPC} \approx \text{path hash table evolution time} + ST_{DS} \qquad (7)$$

Path hash table evolution time is very negligee time. So, this equal to ST_{DS}.

Query Messages (QM): When considering the flooding and MBFS cases, the query message eh generated at hop h is given by:

$$eh = \begin{cases} p.G'_0(1), & \text{for } h = 1 \\ p.G'_1(1).C_{h-1} & \text{for } h \geq 2 \end{cases} \qquad (8)$$

When considering the RW case, the number of query messages for each hop keeps fixed as k, i.e., the number of walkers:

$$eh = k \times TTL \qquad (9)$$

The calculation of query messages for DS depends on h and n. The query messages eh generated at hop h for DS can be written as:

$$eh = \begin{cases} p.G'_0(1), & \text{for } h = 1 \\ p.G'_1(1).C_{h-1} & \text{for } h \geq 2 \\ C_n & \text{for } h > n \end{cases} \qquad (10)$$

The calculation of query messages for SRPC depends on nodes path hash table, path hash table updation and DS.

If node's path hash table has the path information of requested source then eh of SRPC becomes:

$$eh = 1 \qquad (11)$$

If node's path hash table do not have the path information of requested source then eh of SRPC becomes:

$$eh = eh_{DS} \qquad (12)$$

RESULTS AND DISCUSSION

In this study, we show the numerical results of performance evaluation. We show the effectiveness of our SRPC algorithm and the effects of parameters n and p.

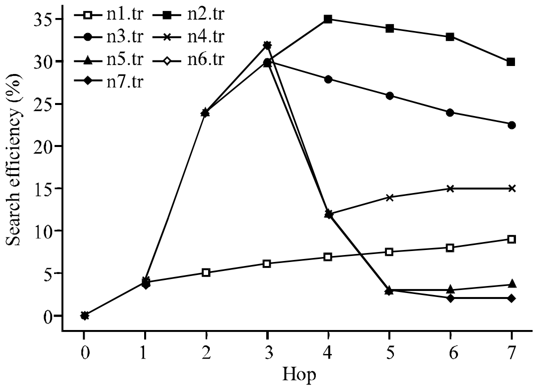

Fig. 1: SE versus hop count when p is set as 1 and n is changed from 1-7. Random topology with N = 10000. When n is set as 2, SRPC gets the best performance for almost all hop counts

Performance of SRPC: This study describes, how the decision threshold n of DS would affect the system performance. Due to space constraints, we only show the result when p is set as 1. The case n = 1 is analog to RW with K(no of walks) equal to the number of first neighbors. The case n = 7 is equal to the flooding. As Fig. 1 shows DS with n = 7 sends the query messages aggressively in the first three hops and gets good Search Efficiency (SE).

However, the performance degrades rapidly as the hop increases. This is because the cost grows exponentially with the path length between the query source and the target. On the contrary, SE of RW is better than that of the flooding when the hop is 5-7. When n is set as 2, SRCP gets the best SE for almost all hop counts. The Fig. 1 shows that a good choice of parameter n can help SRCP to take advantage of different contexts under which each search algorithm performs well.

Performance evaluation results of search time: In this study, N(no of nodes) is set as 10,000, R(replication ratio) is set as 0.01 and TTL(maximum hop length) is set as 7. Similar results can be obtained when the parameters are set as other values.

The walkers K for RW are set as 1 and 32. The decision thresholds n are set as 2, 3 and 7 and p is set as 1. TTL is set as 7 in this case, thus DS with n = 7 are equal to flooding. SRPC with large n always gets the short ST because it always covers more vertices. On the contrary, RW with K = 1 always gets the longest ST since its coverage is only incremental by one at each hop. When K is set as 32, its coverage is enlarged and ST can be improved. However, SRPC still performs better than RW with 32 walkers even when n is set as only 2. Note that

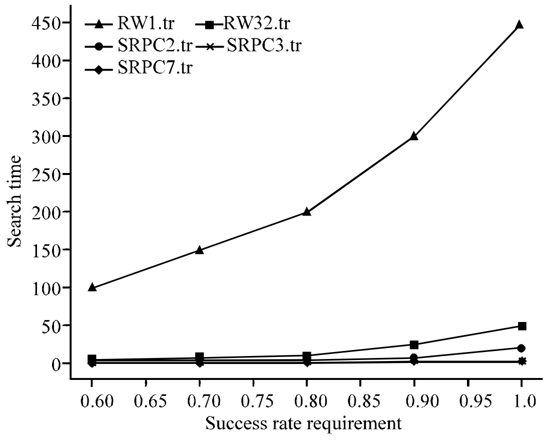

Fig. 2: ST versus SR requirement. R is set as 0.01 in this case. The walkers K for RW are set as 1 and 32, respectively. The n of SRPC are set as 2, 3 and 7 and p is set as 1. TTL is set as 7 in this case, thus the SRPC with n = 7 are equal to flooding

when n is set as 3, SRPC performs as well as that with n = 7, i.e., the flooding, while does not generate as many query messages. In summary, SRCP with n = 2 and p = 1 would get the best SE and significantly improve ST in this case. While increasing n to 3, although SE is a little degraded, the shortest ST is obtained in Fig. 2.

Scalability: In order to validate the scalability of our SRPC algorithm, we show the search efficiency for different number of nodes in Fig. 3. Nodes N are set as 10,000, 50,000, 100,000 and 500,000, respectively. The replication ratio R is set as 0.01 and TTL is set as 7. The following figure shows that our SRPC algorithm always performs better than flooding and RW in spite of the number of nodes.

Bandwidth consumtion: Figure 4 gives us the cost of the various values of Query Message Generation (QMG) in terms of average bandwidth. Along the x-axis we vary the hop count. Recall that QMG gives us the ratio of queries to joins/leaves in the network; the default value observed in Gnutella is roughly 10. We do not vary QueryUpdateRatio because it has the same types of effects as varying QMG, to a lesser degree. We see huge cost savings from the SRCP techniques, especially as QMG increases.

At hop count = 2, SRPC consume only 39% of the bandwidth. From the graph we can conclude that SRPC averagely 5 time better than DS,100 times better than flooding and 250 times better than RW. If hop count increases SRCP bandwidth consumption is out perform than flooding, RW and DS.

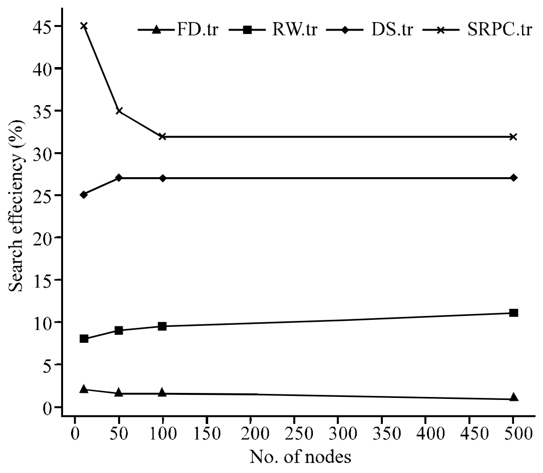

Fig. 3: Search efficiency for different number of nodes N in the network. This figure shows the scalability of the SRPC algorithm

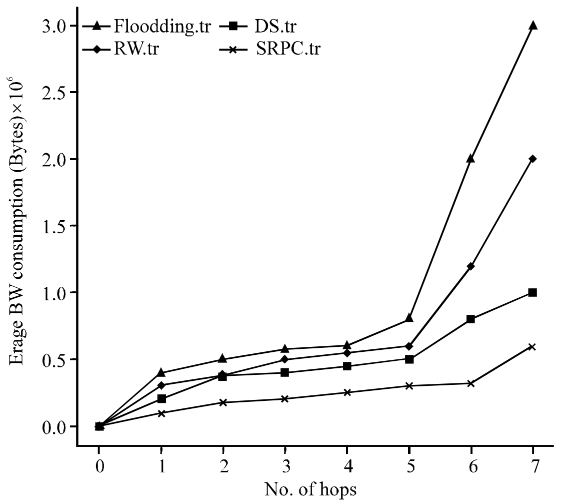

Fig. 4: SRPC Bandwidth consumption analysis with flooding, RW and DS

CONCLUSION

In this study, the proposed SRPC is a generalized form of the DS. It was proposed to overcome the various disadvantages from DS and it performs well in different contexts. The performance of SRPC was analyzed from various metrics such as success rate, search time, number of query hits and number of query messages, query efficiency and search efficiency. Numerical analysis shows SRPC has improved the efficiency, search cost and performance. From Numerical analysis we can conclude that SRPC 5 times better than DS, 125 times better than BFS and 275 times better than RW in power-law graphs. The further improvements needed for validating the available paths and short path detection from search result path information.

REFERENCES

Gang, C., C.P. Low and Z. Yang, 2008. Enhancing Search Performance in Unstructured P2P Networks Based on Users' Common Interest. IEEE. Trans. Parallel and Distributed Syst., 19 (6): 821-836.

Jiang, S., L. Guo, X. Zhang and H. Wang, 2008. LightFlood: Minimizing redundant messages and maximizing scope of peer-to-peer search. IEEE. Trans. Parallel Distributed Syst., 19 (5): 601-614.

Kalogeraki, V., D. Gunopulos and D. Zeinalipour-Yazti, 2002. A local search mechanism for peer-to-peer networks. Proceedings of ACM CIKM International Conference on Information and Knowledge Management Virginia, Nov. 4-9. USA, pp: 300-307.

Kan, G., 2001. Gnutella, Peer-to-Peer Harnessing the Power of Disruptive Technologies. O'Reilly Publications, Chapter 8, pp: 94-122.

Lafferty, J., A. McCallum and F. Pereira, 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. Proceedings of 18th International Conference on Machine Learning (ICML), June 28 to July 1. Williamstown, MA, USA, pp: 282-289.

Lv, Q., S. Ratnasamy and S. Shenker, 2002a. Can heterogeneity make gnutella scalable? Proceedings of First International Workshop on Peer-to-Peer Systems (IPTPS), Mar. 7-8. Cambridge, MA, USA, pp: 94-103.

Lv, Q., P. Cao, E. Cohen, K. Li and S. Shenker, 2002b. Search and replication in unstructured peer-to-peer networks. Proceedings of 16th International Conference on Supercomputing, June 22-26. New York, USA, pp: 84-95.

Lv, Q., P. Cao, E. Cohen, K. Li and S. Shenker, 2002c. Search and replication in unstructured peer-to-peer networks. Proceedings of 16th ACM International Conference on Supercomputing (ICS), June 22-26th. New York, USA, pp: 84-95.

Milojicic, D., V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu, 2003. Peer-to-Peer Computing. 3rd July. Technical Report HPL-2002-57(R.1), HP.

Newman, M.E.J., S.H. Strogatz and D.J. Watts, 2001. Random Graphs with Arbitrary Degree Distribution and Their Applications. Working Papers, 00-07-042, Santa F Institute.

Rasti, A.H., D. Stutzbach and R. Rejaie, 2006. On the long-term evolution of the two-tier gnutella overlay. Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM), Apr. 23-29, Barcelona, pp: 1-6.

Ritter, J., 2001. Why Gnutella Can't Scale. No, Really. http://www.monkey.org/~dugsong/mirror/gnutella.htm.

Rowstron, A. and P. Druschel, 2001. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Procedings of 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Nov. 12-16. Heidelberg, Germany, pp: 329-350.

Sripanidkulchai, K., 2001. The Popularity of Gnutella Queries and Its Implications on Scalability, white paper Featured on O'Reilly's www.openp2p.com.

The Gnutella, 2003. http://gnutella.wego.com.

Tsungnan, L., P. Lin, H. Wang and C. Chen, 2009. Dynamic search algorithm in unstructured peer-to-peer networks. IEEE Trans. Parallel and Distributed Syst., 20 (5): 654-666.

Yang, B. and H. Garcia-Molina, 2002. Improving search in peer-to-peer networks. Proceedings of 22nd International Conference on Distributed Computing Systems, Vienna, 2-5th July. Austria, pp: 5-14.

Zhuang, Z., Y. Liu, L. Xiao and L.M. Ni, 2003. Hybrid periodical flooding in unstructured peer-to-peer networks. Proceedings of 32nd International Conference on Parallel Processing (ICPP) Kaohsiung, 6-9th Oct. Taiwan, pp: 171-178.