# Dynamic Search Algorithm in P2P Networks

Prabhudev.S.Irabashetti
*M.tech Student,UBDTCE, Davangere,*

**Abstract- Designing efficient search algorithms is a key challenge in unstructured peer-to-peer networks. Flooding and Random Walk (RW) are two typical search algorithms. Flooding searches aggressively and covers the most nodes. However, it generates a large amount of query messages and thus does not scale. On the contrary, RW searches conservatively. It only generates a fixed amount of query messages at each hop but would take longer search time. Proposed Dynamic Search (DS) algorithm, which is a generalization of Flooding and RW. DS takes advantage of various contexts under which each previous search algorithm performs well. It resembles flooding for short-term search and RW for long-term search. Moreover, DS could be further combined with knowledge-based search mechanisms to improve the search performance. Analyzing the performance of DS based on some performance metrics including the success rate, search time, query hits, query messages, query efficiency, and search efficiency. Numerical results show that DS provides a good tradeoff between search performance and cost. On average, DS performs about 25 times better than flooding and 58 times better than RW in power-law graphs, and about 186 times better than flooding and 120 times better than RW in bimodal topologies.**

## I. INTRODUCTION

In unstructured Peer-to-Peer (P2P) networks, each node does not have global information about the whole topology and the location of queried resources. Because of the dynamic property of unstructured P2P networks, correctly capturing global behavior is also difficult. Search algorithms provide the capabilities to locate the queried resources and to route the message to the target node. Thus, the efficiency of search algorithms is critical to the performance of unstructured P2P networks. Previous works about 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, either generating too much load on the system, or not meeting user's requirements. Flooding, which belongs to BFS-based method, is the default search algorithm for Gnutella network. By 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. 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. It produces considerable query messages even when the resource distribution is scarce. The search is especially inefficient when the target is far from the query source because the number of query messages would grow exponentially with the hop counts. Fig. 1.1 illustrates the operation of flooding. The link degree of each vertex in this graph is 4. If the network grows unlimited from the query source, the number of query messages generated by flooding at each hop would be 4, 12, 36. . respectively. If the queried resource locates at one of the third neighbors, it takes 4 + 12 + 36 = 52 query messages to get just one query hit.

On the other hand, Random Walk (RW) is a conservative search algorithm, which belongs to DFS-based methods. By RW, the query source just sends one query message (walker) to one of its neighbors. If this neighbor does not own the queried resource, it keeps on sending the walker to one of its neighbors, except for the one the query message comes from, and thus the search cost is reduced. The main drawback of RW is the long search time. Since RW only visits one node for each hop, the coverage of RW grows linearly with hop counts, which is slow compared with the exponential growth of the coverage of flooding. Moreover, the success rate of each query by RW is also low due to the same coverage issue. Increasing the number of walkers might help improve the search time and success rate, but the effect is limited due to the link degree and redundant path. As an example shown in Fig. 1.1, RW can only visit 12 vertices of second neighbors even when the number of walkers is set as 32. Certainly, the search is inefficient because 32 walkers only visit 12 vertices at the second hop.

The proposed Dynamic Search (DS) algorithm, which is a generalization of Flooding and RW. DS overcomes the disadvantages of flooding and RW and takes advantage of different contexts under which each search algorithm performs well. The operation of DS resembles flooding for the short-term search and RW for the long-term search. In order to analyze the performance of DS, we apply the random graphs as the models of network topologies and adopt the probability generating functions to model the link degree distribution. We evaluate the performance of search algorithms in accordance with some

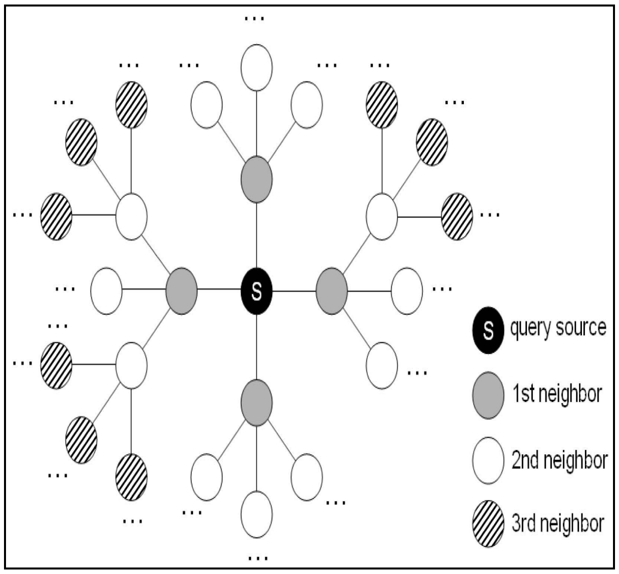


Fig. 1.1: A simple scenario of P2P network to demonstrate the operation of Flooding and Random Walk.

performance metrics including the success rate, search time, number of query hits, number of query messages, query efficiency, and search efficiency. Simulation experiments are performed in a dynamic P2P networking environment in order to collect convincing results for algorithm evaluations. The factors considered include the network topology, link degree distribution, peer's joining and leaving, and querying behavior as well as the activity of file sharing. Our dynamic network model is constructed based on these factors that strongly reflect the real measurement studies. Numerical results show that DS could provide a good tradeoff between search performance and cost. On average, DS performs about 25 times better than flooding and 58 times better than RW in power-law graphs, and about 186 times better than flooding and 120 times better than RW in bimodal topologies.

## *1.1 Problem Statement*

In unstructured peer-to-peer networks, each node does not have global information about the whole topology and the location of queried resources. Because of the dynamic property of unstructured P2P networks, correctly capturing global behavior is also difficult. So all the search algorithms used in unstructured peer to peer networks, does not provide efficiency. Thus, the efficiency of search algorithms is critical to the performance of unstructured P2P networks.

## *1.2 Aim of the Project*

The main aim of the project work is to design Dynamic Search (DS) algorithm, which is a generalization of flooding and RW. DS overcomes the disadvantages of flooding and RW and takes advantage of different contexts under which each search algorithm performs well. The operation of DS resembles flooding for the short-term search and RW for the long-term search.

This methodology will make the network design simpler and more efficiently, which is a generalization of flooding and RW. In order to analyze the performance of DS, we apply the random graphs as the models of network topologies and adopt the probability generating functions to model the link degree distribution. We evaluate the performance of search algorithms in accordance with some performance metrics including the success rate, search time, number of query hits, and number of query messages, query efficiency, and search efficiency.

It considers the probabilistic function based on the information learned from the past experiences which are known as Knowledge-Based Dynamic Search, with respect to each search target, search time, and local topology information. Thus, a node has more information to intelligently decide how many query messages to send and to which peers these messages should be forwarded.

### *1.3 Proposed Algorithm*

DS is designed as a generalization of flooding and RW. There are two phases in DS. Each phase has a different searching strategy. The choice of search strategy at each phase depends on the relationship between the hop count h of query messages and the decision threshold n of DS. Here n is the threshold value and h is the hop count.

Phase1. When h <= n

At this phase, DS acts as flooding. The number of neighbors that a query source sends the query messages to depends on the predefined transmission probability p. If the link degree of this query source is d, it would only send the query messages to d. p neighbors. When p is equal to 1, DS resembles flooding.

Phase2. When h > n

At this phase, the search strategy switches to RW. Each node that receives the query message would send the query message to one of its neighbors if it does not have the queried resource. Assume that the number of nodes visited by DS at hop h = n is the coverage, and then the operation of DS at that time can be regarded as RW with an walkers. However, there are some differences between DS and RW when we consider the whole operation.

## II. DATA FLOW DIAGRAM

Data flow models are an intuitive way of showing how data is processed by a system. At the analysis level, they should be used to model the way in which data is processed in the existing system. The notations used in these models represents functional processing, data stores and data movements between functions. Data flow models are used to show how data flows through a sequence of processing steps. The data is transferred at each step before moving on to the next stage. These processing steps or transformations are program functions when data flow diagrams are used to explain a software design.

The Data Flow Diagram (DFD) is a graphic tool used for expressing system requirements in a graphical form. The DFD also known as the “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that to become program in system design.

Thus DFD can be stated as the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. The DFD consists of series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system. A DFD describes what data flow is rather than how they are processed, so it does not depend on hardware, software, data structure or file organization.

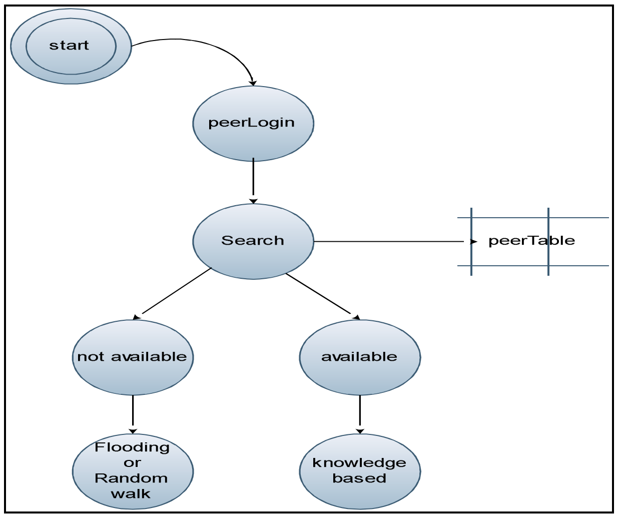


Fig. 2.1: Data Flow Diagram

Fig. 2.1 shows the Data Flow Diagram in this first each peers logins to the network, when there is a query for searching the resource then the peer first checks its peer table also called as routing table. If the resource is found or available then the knowledge based table is updated, so that next time when same resource is queried by the peer it can be searched quickly. If resource is not available then it is searched using the either of the search techniques.

*2.1. Modules:*

a) Peer Construction

b) Searching

c) Knowledge based Searching

*2.2. Peer Construction*

In this module, we construct a topology structure. While getting each of the nodes, their associated port and ip address is also obtained. For successive nodes, the node to which it should be connected is also accepted from the user. While adding nodes, comparison will be done so that there would be no node duplication. Then we identify which peer is going to request for a file.

Fig. 2.2 shows peer construction data flow diagram. Here the peer first logs in to the network, by entering the port number and the ip address. It is authenticated using the Knowledge Search Database. Then if the ip address and the port number matches then the search screen is displayed and also the peer info table in the Knowledge Search database is updated, here the peer status column is set to ON so that the same peer cannot login again.

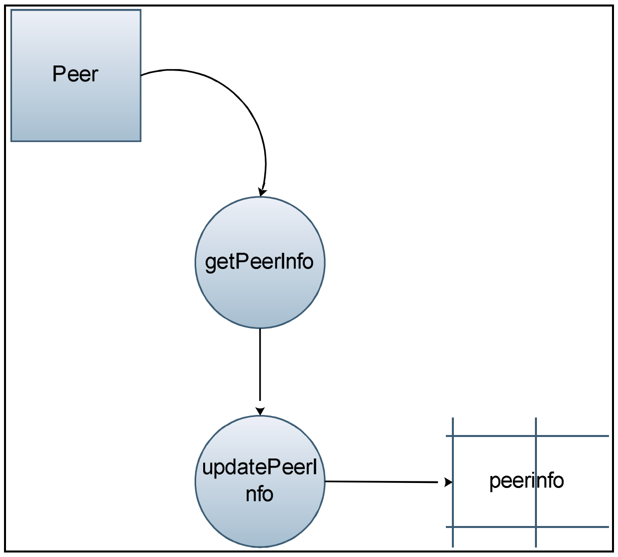


Fig. 2.2: Peer construction data flow diagram

*2.3. Searching*

In this module a peer can search a file.

Phase1. When h <= n

At this phase, DS acts as flooding or MBFS. The number of neighbors that a query source sends the query messages to depends on the predefined transmission probability p. If the link degree of this query source is d, it would only send the query messages to d. p neighbors. When p is equal to 1, DS resembles flooding. Otherwise, it operates as MBFS with the transmission probability p.

Phase2. When h > n

At this phase, the search strategy switches to RW. Each node that receives the query message would send the query message to one of its neighbors if it does not have the queried resource. Assume that the number of nodes visited by DS at hop h ¼ n is the coverage cn, and then the operation of DS at that time can be regarded as RW with cn walkers. However, there are some differences between DS and RW when we consider the whole operation.

In the Fig. 2.3 after the peer login the search screen is displayed, user can enter the name of the file to be searched. First the peer searches in its own peer if the file is not available in the local system the it searches in the other system. It will ask the user to enter the number of nodes to be searched, if the entered value is less than the threshold then the searching method will be flooding or else it will be random walk.

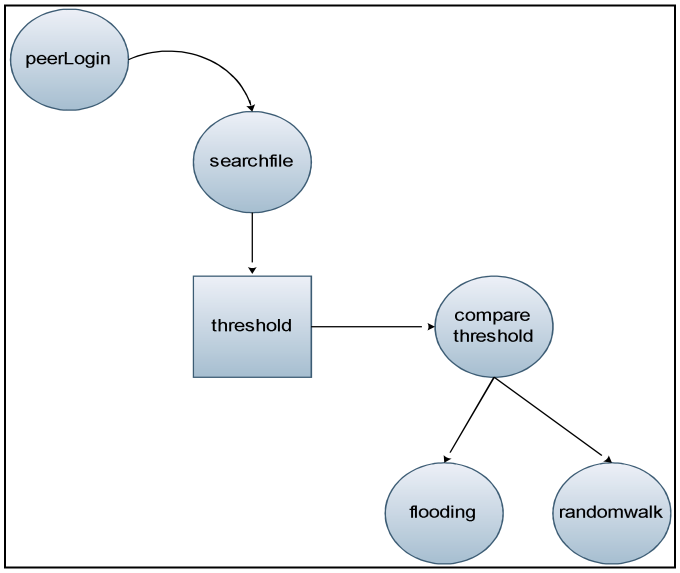


Fig. 2.3: Data flow diagram for searching the resource

## *2.4 Knowledge Based Searching*

In this module we search the file more intelligently. Here we search the peer routing table whether the file is already searching or not. If we already search means it relays the query more intelligently to the corresponding peer. Some knowledge-based search algorithms, including APS, biased RW, RI, local indices, and intelligent search, are applicable to combine with our DS algorithm, and any training or caching operations are benefit from our DS algorithm as well. In this section, we present the generic scheme to incorporate these knowledge-based search algorithms with our DS algorithm. 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. Thus, a node has more information to intelligently decide how many query messages to send and to which peers these messages should be forwarded. Take APS as an example. The peer applying APS search builds a probability table for each neighbor and each object. It consistently refines its probability table by the search experiences. If a search query for some object delivers to a certain neighbor successfully, the probability entry corresponding to that neighbor and object is increased. If the search fails finally, it will decrease the probability entry. In accordance with APS, when a certain node receives a hit from peer i, it adds 10 points for the entry of peer i; if peer I fails to respond the hit to that node, the node subtracts 10 points for the entry of peer i.

Knowledge Based Searching: the figure Fig. 3.1 shows the Knowledge Based Searching data flow diagram

## III. DETAILED DESIGN

The introduction of structured programming in the 1960's and 70's brought with it the concept of Structured Flow Charts. In addition to a standard set of symbols, structured flow charts specify conventions for linking the symbols together into a complete flow chart.

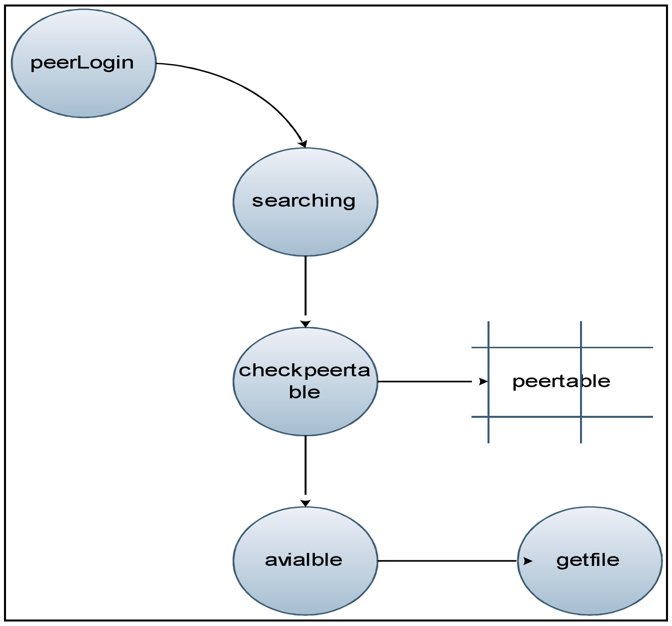


Fig. 3.1: Shows the Knowledge Based searching data flow diagram

The structured programming paradigm evolved from the mathematically proven concept that all problems can be solved using only three types of control structures.

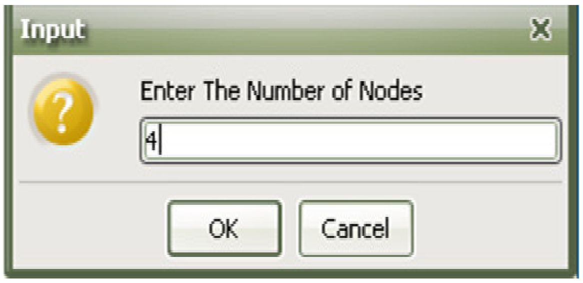
- Sequence
- Decision (Selection)
- Iterative (or looping)

A Structure Chart (SC) in software engineering and organizational theory is a chart, which shows the breakdown of the configuration system to the lowest manageable levels. It is used to show the hierarchical arrangement of the modules in a structured program. Each rectangular box represents a module. The names of the modules are written inside the box. An arrow joins two modules that have an invocation relationship.

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. The design of the system is perhaps the most critical factor affecting the quality of the software. Here we build the System Block Diagram that will be helpful to understand the behavior of the system. Here we divide problem into modules. Data flow Diagrams show flow of data between/among modules.

## IV. RESULTS

Following are the sample screen shots of the project. First the clients must register to the server, so at the beginning the number of peer connected to the server must be known and the value must be entered in the Fig. 4.1.



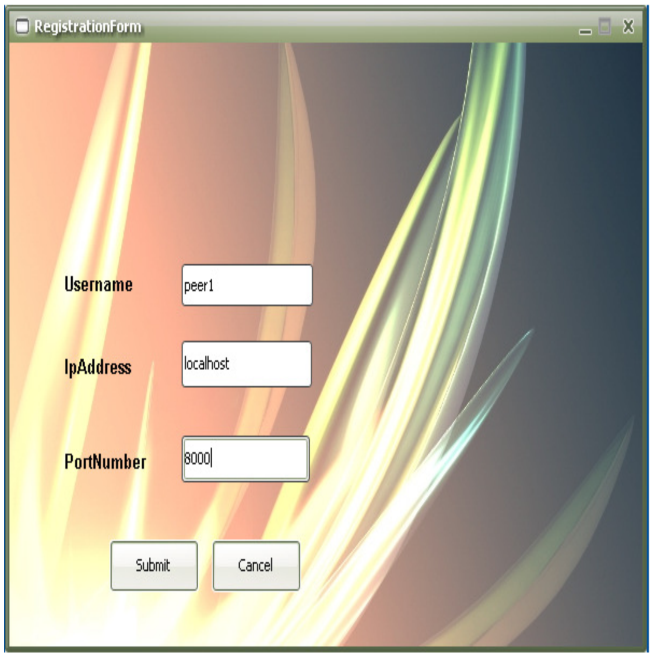
Enter The Number of Nodes

4

Fig. 4.1: To enter the number of nodes

Fig. 4.2 shows the registration form, all the peers must be registered to the server, the first field entered is the Username it's the name of the user of that peer, the second field is the Ip Address it's the Ip address of the peer which is connecting, even the hostname of the peer can be entered and last field is the Port Number.

When the peer loges on to its machine at that time it must connect  to the server the by entering the Peer Name and the Port Number so as to authenticate the user, the login form looks like Fig. 4.3. Then if the information is entered correctly then the peer is connect to the server and search screen popup shown in Fig. 4.4. In the text filed area the user must enter the file to be searched, then after submitting the user will get the name of the peer where the file is available if not a popup appears stating that the file in not there in the network.

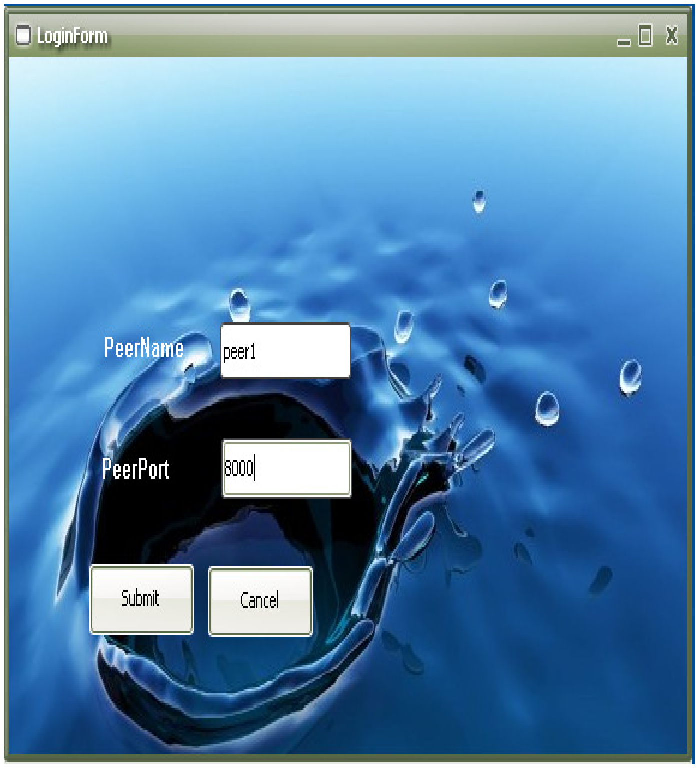


Username peer1

IpAddress localhost

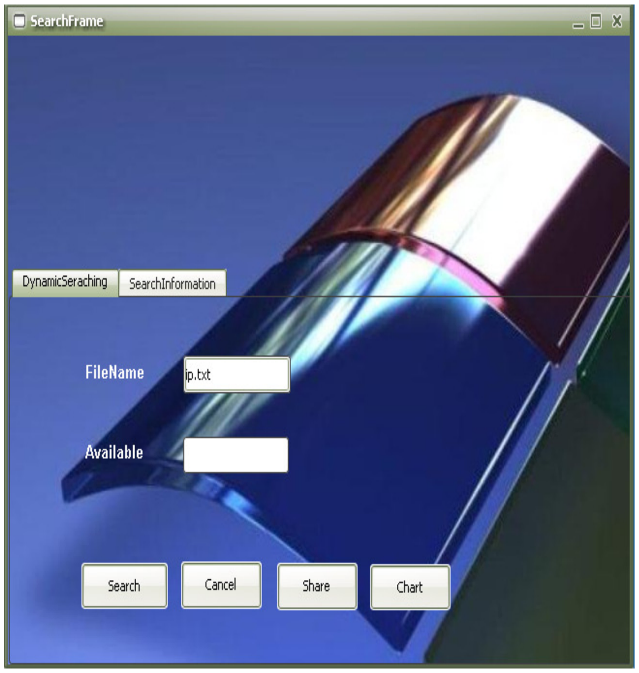
PortNumber 8000

Fig. 4.2: Registration Form



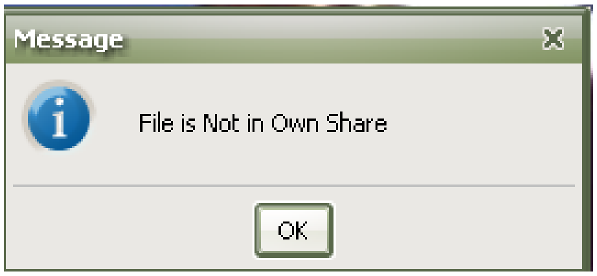
PeerName: peer1
PeerPort: 8000

Fig. 4.3: Login Form



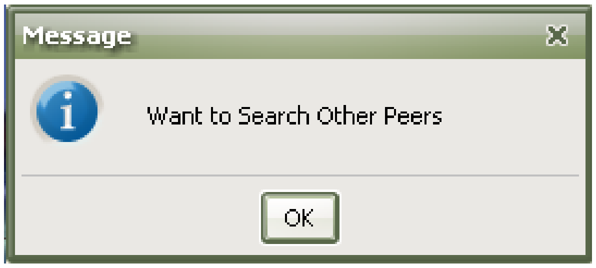
FileName: ip.txt

Fig. 4.4: Search Frame


Message

File is Not in Own Share

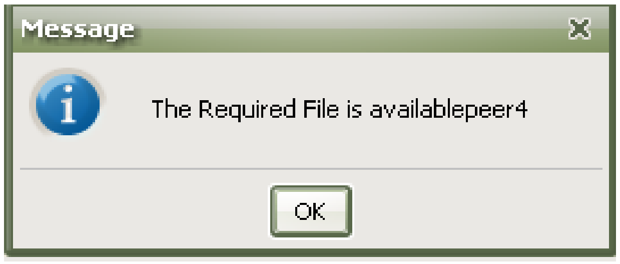
Fig. 4.5: File is not found in the searching peer


Message

Want to Search Other Peers

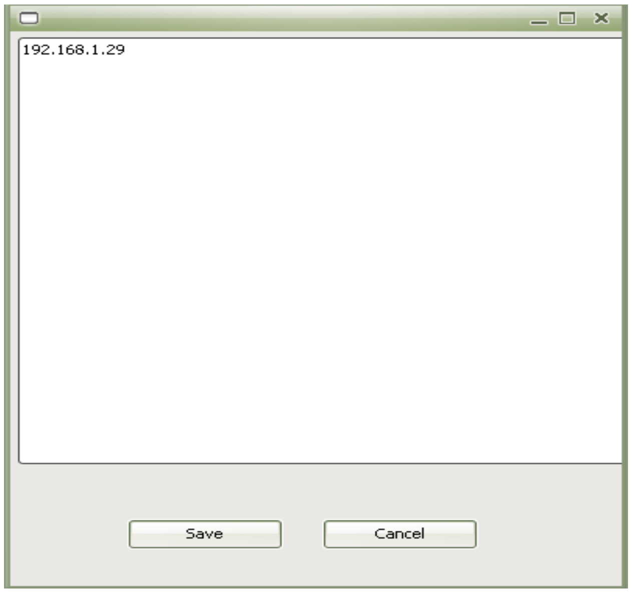
Fig. 4.6: To search in other peers

Fig. 4.2 to 4.7 shows the different screen which are popped up while searching the file, here first the file is searched in its own peer if available then displayed if not it is then searched in the knowledge based search, if that is not searched previously then the file is search in other peers depending upon the threshold value entered. If the file is found then the file


Message

The Required File is availablepeer4

Fig. 4.7: Required file is available in peeer4



```
192.168.1.29
```

Fig. 4.8: To display file content

content is displayed as shown in Fig. 4.8. the user has a option to save the file on his local machine. This project works for the files having .txt and .java as its extension.

Fig. 4.9 and 4.10 shows the sample screen shots of the chart which are generated after searching the file, which clearly shows the time comparison of different searching techniques. Fig. 4.11 shows search information details and also it is used if then next time the same file is being searched. It gives the name of the file being searched, alos the available peer name where it is present, the searching algorithm and the time required to search the file.

## V. CONCLUSION

The Dynamic Search algorithm is a generalization of the Flooding and RW. It resembles flooding for the short term search and RW for the long-term search. We analyze the

performance of DS based on some metrics including the success rate, search time, number of query hits and number of query messages, query efficiency and search efficiency.

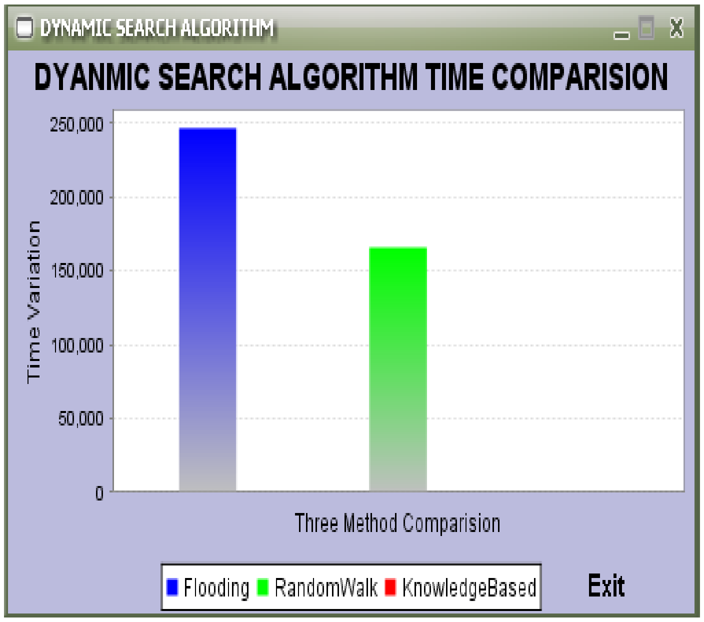


Fig. 4.9: Chart to display the operation of Dynamic search algorithm time comparison of Flooding and Random Walk

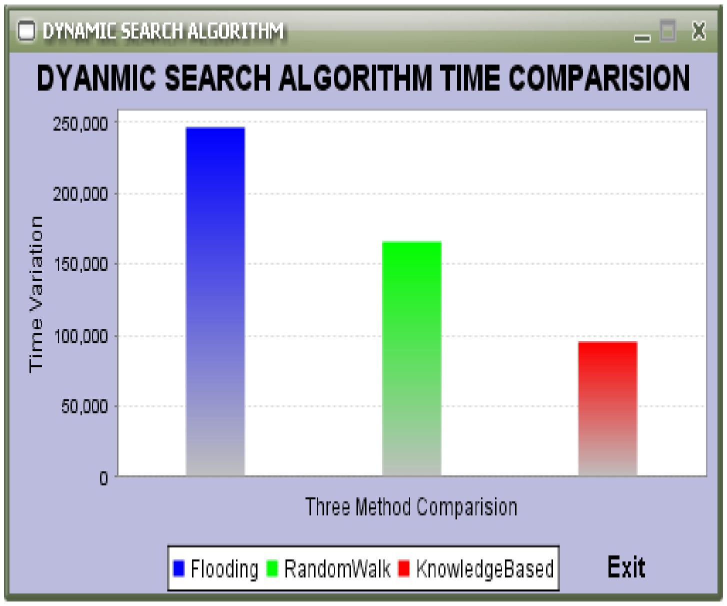
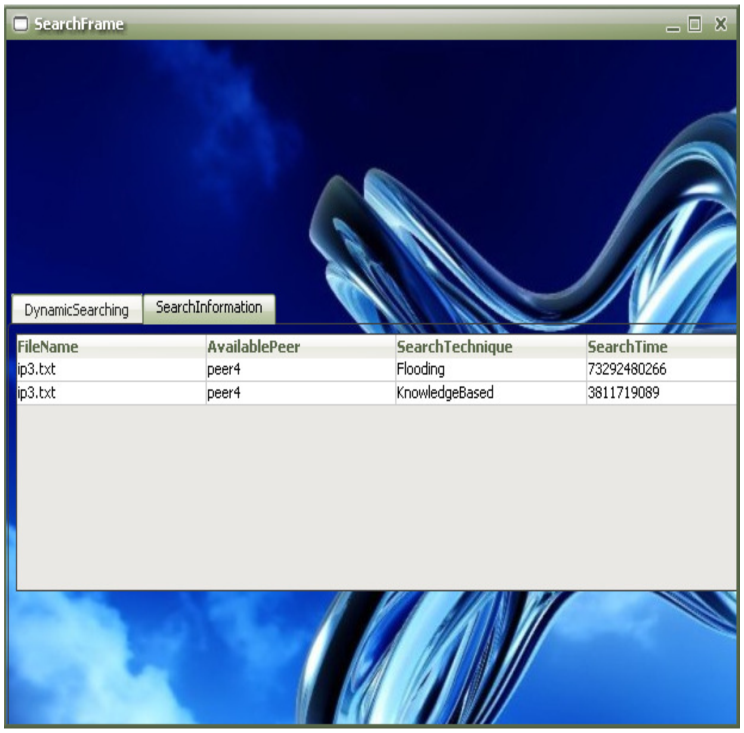


Fig. 4.10: Chart to display the operation of Dynamic search algorithm time comparison of Flooding , Random Walk and Knowledge Based searching

DS always performs well, when combined with knowledge-based search algorithms, its search performance could be further improved. DS reduces search time and provides a good tradeoff between the search performance and cost.

DS overcomes the disadvantages of Flooding and RW technique. Knowledge-based search algorithms take advantage of the knowledge learned from previous search results thus increase the speed of DS.



| FileName | AvailablePeer | SearchTechnique | SearchTime |
|---|---|---|---|
| ip3.txt | peer4 | Flooding | 73292480266 |
| ip3.txt | peer4 | KnowledgeBased | 3811719089 |

Fig. 4.11: Search Info table after  Searching

## REFERENCES

[1] Vicent Cholvi, Pascal Felber and Ernst Biersack, “Efficient Search in Unstructured Peer-to-Peer Networks” Institut EURECOM Sophia-Antipolis (France).
[2] Qin Lv, Pei Cao and Edith Cohen, “Search and Replication in Unstructured PeertoPeer Networks” Dept. of Computer Science Princeton University.
[3] Christos Gkantsidis, Milena Mihail and Amin Saberi, “Hybrid Search Schemes for Unstructured Peer-to-Peer Networks” Institute for Computational and Mathematical Engineering and Management Science and Engineering Department Stanford University Stanford, CA, 94305, USA.
[4] Xiuqi Li and Jie Wu, “Cluster-based intelligent searching in unstructured peer-to-peer networks” This paper appears in Distributed Computing Systems Workshops, 2009. 25th IEEE International.
[5] Xiuguo Bao, Binxing Fang, Mingzhen Hu and Binbin Xu, “Heterogeneous Search in Unstructured Peer-to-Peer Networks” IEEE Distributed Systems Online 1541-4922 2005 Published by the IEEE Computer Society Vol. 6, No. 2; 2005
[6] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, “Sampling Techniques for Large, Dynamic Graphs,” Proc. Ninth IEEE Global Internet Symp. (Global Internet ’06), Apr. 2008.
[7] A.H. Rasti, D. Stutzbach, and R. Rejaie, “On the Long-Term Evolution of the Two-Tier Gnutella Overlay,” Proc. Ninth IEEE Global Internet Symp. (Global Internet ’06).