Random Walk with Jumps: A New Query Search Method Based on Analysing Gnutella Protocol

Kholoud Althobaiti
Computer Sciences Department
Taif University Taif, Saudi Arabia
kholod.k@tu.edu.sa

Sara Jeza Alotaibi
Computer Sciences Department
Taif University, Saudi Arabia
sara.alotaibi@tu.edu.sa

Hend Alqahtani
Computer Sciences Department
Umm Al-Qura University
Jeddah, Saudi Arabia
h.h.alqahtany@gmail.com

Abstract—**The measurement of a search algorithm for unstructured P2P network centres on the number of nodes not receiving their requested files (number of failures) and the number of hops per query. Most current search algorithms are unable to guarantee the success of the query. This study involves a comparison of the strengths and weaknesses of three algorithms of Gnutella P2P protocol, namely Flood, Random Walk, and Random Walk with Neighbours Table. Based on this comparison, a new query search method—referred to as Random Walk with Jumps—is proposed. The experiment proves that the proposed algorithm can obtain a better result with a small number of failures and a minimum number of hops.**

***Keywords*—Peer-to-peer, Gnutella, Query, Method, Network, Analysing.**

I. INTRODUCTION

Peer-to-peer systems (P2P) have emerged as a big social and technical event over the past 15 years [1]. They have come up with an infrastructure for communities that share CPU and/or storage space (for example, Gnutella), or that support collaborative environments [9]. Two key reasons have promoted the rapid growth of such systems, namely the low cost and the large number of storage resources on the one hand, and the increased network connectivity on the other hand [9]. Therefore, the P2P network has been gaining in popularity over recent years.

One of the challenges in P2P networks is searching the content of nodes (files). In pure P2P systems, individual computers communicate directly with ne another and share information and resources without using dedicated servers. In essence, this is more like a self-organised network of independent entities. Gnutella protocol specifically is an open, decentralised group membership and search protocol, primarily used for file-sharing. It has been recognised as a popular file-sharing protocol in P2P networks [2]. These features have motivated us to compare three search methods of the P2P system Gnutella, namely Flood, Random Walk and Random Walk with Neighbours Table.

The file-sharing system in Gnutella network operates as follows:

1- A user (Node A) starts with a networked computer that runs one of the Gnutella clients. The user then connects to another Gnutella networked computer (Node B). Subsequently, Node A announces its existence to Node B.

2- Node B then announces to all its neighbouring nodes that Node A is existent.

3- Once the rest of the nodes are aware of the existence of Node A, the user at Node A then is positioned to query the data shared across the network. [6]

This paper presents the measurements and analysis of the three methods of Gnutella (Flood, Random Walk and Random Walk with Neighbours Table). The measurements and the analysis are carried out when the number of nodes is variable. They are driven by two primary questions:

- Which of the three methods has the least average hops and/or number of failures, and which has the highest?
- What is the impact of expanding the network on the success of the query?

In light of the outcomes, this study proposes a new query search method for Gnutella that improves the current code and accordingly makes it more scalable.

The rest of the paper is structured as follows: Section II presents the background and the Literature Review, whilst Section III describes the comparison of the three methods, which helps to answer the questions introduced earlier. Section IV introduces the new query search method for Gnutella. The results and testing are given in Section V, whilst Section VI presents the conclusion and suggestions for future work.

II. BACKGROUND AND LITERATURE REVIEW

The aim of this paper is centred on identifying the challenges that still exist in Gnutella P2P query methods. There are many studies concerned with improving the Gnutella network: in 2003, researchers Tsungnan & Hsinping concluded that the Flooding algorithm generates the best performance in terms of Search Responsiveness, but its Query Efficiency is low due to a huge number of redundant messages. The Random Walk algorithm enjoys high query efficiency, but nonetheless suffers from low search responsiveness [14]. From other points, the study concludes that one of the advantages of Random Walk over Flooding in unstructured overlays is that the former provides more precise control over the number

978-1-908320-50/6/$31.00 ©2015 IEEE

of overlay nodes that is visited to satisfy a query. [15]

However, this study differs significantly from past works, both in purpose and detail. The researchers used an existing Gnutella code to analyse the current query methods in order to assess their overall ability to satisfy the file search requirements. Then, based on the results, the researchers developed an improved query method with a small number of failures and a minimum number of hops.

The study started by examining the Gnutella code [3], with the reason behind using this source owing to the fact that it is open source and is written in Java programming language, which is the preference amongst researchers.

III. Comparison of Gnutella Methods

A good search method must have a small number of failures and a minimum number of hops. This section shows the comparison of three Gnutella methods, namely Flood, Random Walk and Random Walk with Neighbours Table, using an existing Gnutella open source code to determine the value of each search method.

The comparison has garnered the results of counting the average hops and the number of failures for each of the three methods. The goal of analysing these methods is to answer the questions posed earlier in Section II: Which has the least average hops and/or number of failures, and which has the highest? Moreover, what is the impact of expanding the network on the success of the query? Answering these questions will be pivotal in evaluating the search quality of each of the three Gnutella algorithms.

The study focuses on calculating the number of failures and the average hops resulting from 10,000 runs in an overlay of 10,000–100,000 nodes, taking into account the average of all runs. Some of the values have been fixed in an effort to establish accurate results. The maximum number of neighbours is 10, the number of files is 9,000 (with a maximum number for each node between 5 and 50), and the maximum hops per query is 7.

The rest of this section highlights the advantages and disadvantages associated with each of the three methods discussed as follows.

A. *Flooding Algorithm*

Flooding distributes the file across every graph in a network. The following steps depict how this algorithm works:

- One of the nodes requests a file.
- Each node works as a sender and receiver, except in the case of the first node.
- Each node attempts to send all messages to all its neighbours, with the exception of the source of the message. Therefore, at the very end, the querying node will receive its requested file, as seen in Figure 1.

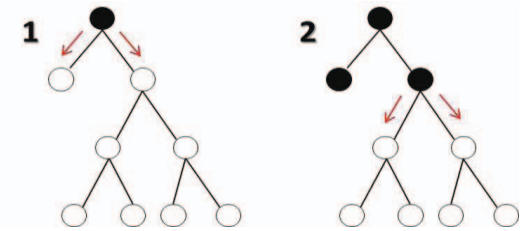

Figure 1: Illustrate Flooding Algorithm

1) *Advantages*

- If the packet can be delivered, it will be.
- Since flooding uses all existing paths, it also uses the shortest path.
- Flooding is easy to implement [4].

2) *Problems*

- Costly: It wastes bandwidth because the message is supposed to arrive at one specific node but instead is sent to all nodes [4].
- Duplicated deliveries may occur, which could drive the algorithm to infinite loop unless certain precautions are taken, such as:
    - Allow each node to keep track of every packet seen, and then forward each of those packets only once.
    - Enforce a network topology without loops (P2P networks like Gnutella do not have topology, so there is no need to take this precaution).

B. *Random Walk Algorithm*

The Random Walk Algorithm is a popular alternative to Flooding. It distributes the file to a Random walker, which chooses a random node to walk to. It is considered the best choice in applications for statistics, Physics and Artificial Intelligent (Bayesian Inference). It is also referred to as Markov Chain [16].

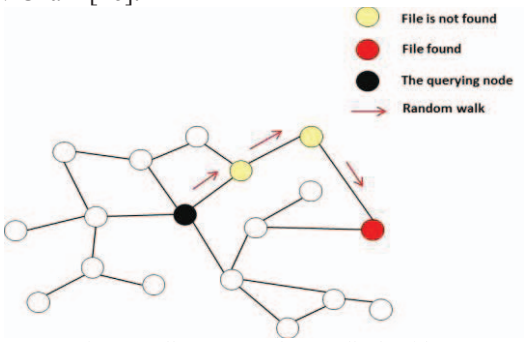

Figure 2: Illustrate Random Walk algorithm.

This Algorithm is explained as shown in Figure 2, with the help of the following steps:

- The querying node sends the number of queries to a randomly selected neighbour. Each one of these queries is referred to as Random walker.
- Each Random walker has a hop count (in this study its value is 7). When a node receives a random walker, it searches for the requested file in the node file list. If the file is not found, the node checks the hop count. If the hop count=>0, it decrements by one and forwards the query to a randomly chosen neighbour. If count=0, the query is not forwarded. On the other hand, if the file is found, the query is not forwarded and a reply is sent to the querying node.

1) *Advantages*

- Equally successful to ordinary Flooding.
- If the file is nearby, it can be considered less costly than Flooding [17–18].
- It does not require any topology information; P2P network does not have any topology network [18-21].
- The performance of the Random walk depends on

choosing the parameters of the Random walk (the number of queries and hop counts). The popularity of a resource is required for the parameters selection module to set their values.

- The number of requests here is fewer than in an ordinary Random Walk [17–21].

2) Problems

- If the file is not found, more messages are generated than in the case of Flooding [18–21].
- Each Random walker needs a time allowance to live or hop count; otherwise, duplicated deliveries could occur.
- Choosing low values for parameters for searching a file with a low popularity estimate would result in a low success rate and high delays, whilst choosing high values of parameters for searching a file with a high popularity estimate would result in excessive overhead [17–18–21].

C. Random Walk with Neighbours Table Algorithm

This is similar to ordinary Random walk; in this method, however, the querying node checks to see if its neighbours' list of files has the desired query within them. If so, that neighbour is walked to if it is not randomly chosen.

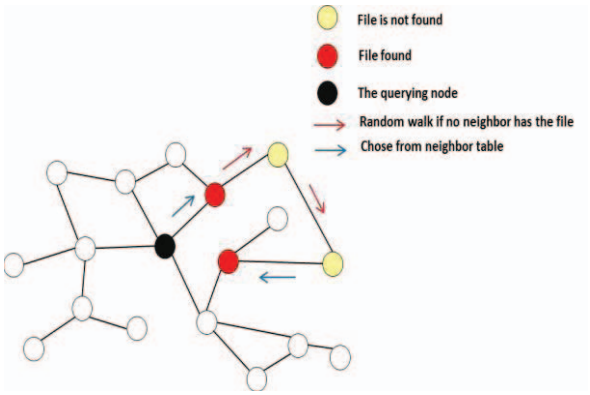

Figure 3: Illustrate Random Walk with Neighbours Table algorithm.

As shown in Figure 3, the algorithm works as follows:

- The method will begin by examining the querying node's list of neighbours. It searches for the requested file in each neighbour's list of files. If the file is found in the file list of any neighbour, the method walks directly to the neighbour. If the file is not found on any list, a neighbour is randomly chosen, and the method then completes another search for the file in the neighbour's list.
- Each one of the randomly generated has a hop count (in this code, its value is 7). When this node receives the request, it searches for the requested file in this node's list of files. If the file is not found, the node checks the hop count. If hop count=>0, it decrements by one and forwards the query to another randomly chosen neighbour. If count=0, the query is not forwarded. On the other hand, if the file is found, the method walks directly to that neighbour.

1) Advantages

- If the requested file is nearby, the method is considered less costly, even when compared with ordinary Random Walk.
- It does not require any topology information; P2P network does not require any [19–20–21].

2) Problems

- If the file is not found, more messages are generated.
- Each random move requires TTL or hop count; otherwise, duplicated deliveries could occur [19–20–21].

D. Results of Analysing Gnutella Methods

In general, the results of the analysis showed that number of failures is the least in Flooding compared to Random Walk and Random walk with Neighbours Table. Although Flooding has the lowest number of failures (strength), as shown in Table 1 and described in Figure 4, at the same time, it also has the highest average of hops (weaknesses). Random Walk shows the lowest average hops (a strength) and a moderate value for the number of failures, as mentioned in Table 2 and Figure 5. The results of the Random Walk with Neighbours Table do not differ significantly from those of the Random Walk method.

TABLE I: ANALYSING RESULT OF THREE METHODS FOR NUMBER OF FAILURES.

	Flood	RW	RW with NT
10000	1153	3733	4489
20000	933	4096	4123
30000	638	3526	3949
40000	583	3581	4018
50000	528	3598	3611
60000	561	2750	3404
70000	419	3560	3669
80000	250	3214	3602
90000	295	3400	3231
100000	387	3221	2817

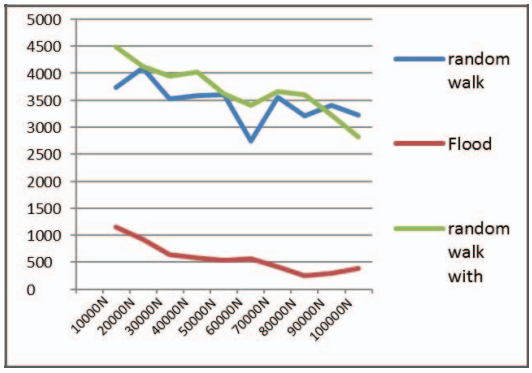

Figure 4: Analysing result of three methods for number of failures

TABLE II: ANALYSING RESULT OF THREE METHODS FOR AVERAGE HOP TIME

	Flood	RW	RW with NT
10000	4.8500	3.9782	3.9151
20000	4.8830	4.1479	4.0993
30000	4.9899	4.4332	4.2190
40000	4.9982	4.4126	4.2155
50000	5.0326	4.4500	4.4166
60000	5.0109	4.9728	4.6250
70000	5.0067	4.5010	4.4335
80000	5.0455	4.7446	4.4353
90000	5.0302	4.6148	4.7522
100000	5.0468	4.7681	5.0731

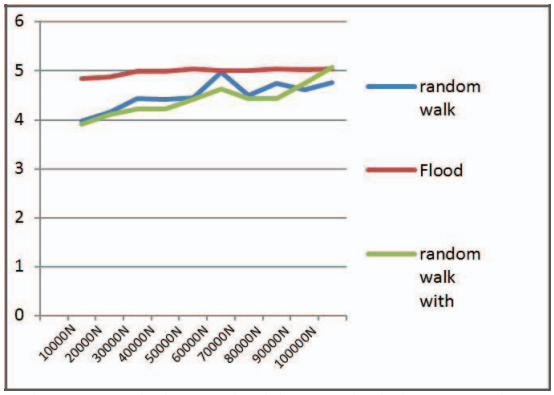

Figure 5: Analysing result of three methods for average hop.

In light of these results, it is noticed that writing a new method should be done in order to minimise the number of failures as in Flooding and, at the same time, minimising the number of requests as in the Random Walk method.

IV. DESIGN NEW SEARCH QUERY METHOD

This paper proposed a new search query method. The so-called Random Walk with Jumps is a popular alternative to Flooding. This algorithm can be considered a Random Walk search algorithm in which the random walk makes jumps. The expected jump length is randomly walked (Figure 6).

A. Algorithm

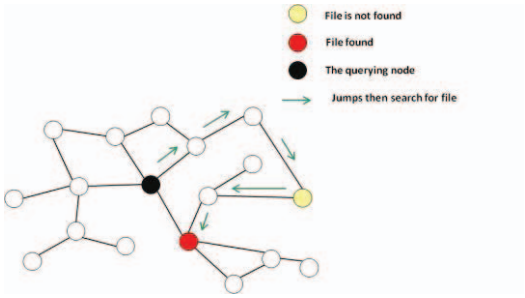

Figure 6: Illustrate Random Walk with Jumps algorithm

The querying node sends the number of queries to a randomly selected neighbour. It then chooses a random length to jump. If the jump length=>0, it passes to another random neighbour and decrements the length by one. If jump=0, it stops passing and searches for the file in the current node. This step then is considered a Single Random Walk.

Each Random walker has a hop count (in this code, its value is 7). When a node receives a Random walker, it searches for the requested file in the node file list. If the file is not found, the node checks the hop count. If hop count=>0, it decrements by one and forwards the query to a randomly chosen neighbour, and accordingly performs another random walk. If count=0, the query is not forwarded. On the other hand, if the file is found, the query is not forwarded and a reply is sent to the querying node.

B. Advantages

- As successful as an ordinary Random walk.
- The jumps reduce focus on a specific area. This advantage provides the ability to cover different areas of node locations and, subsequently, speeds up the search and accordingly reduces network capacity.
- If the file is on a remote area, the new method can be considered less costly than Random walk.
- The method reduces the number of requesting messages, even if there are no files in the graph. The number of requests is smaller than in the ordinary Random Walk.

C. Problems

As with the Random Walk, if low values were chosen for parameters, searching for a file with low popularity estimate would result in a low success rate and high delays, whilst choosing high values of parameters to search with high popularity estimates would result in excessive overheads.

D. Code Description

The code below starts with a conditional statement that allows the querying node to check if there are neighbours in the graph. If the nodes do not have any neighbours, this means the jump is a fail; the value of fail then would change to true.

```
if(neighbors == null || (neighbors.size() <=
  0)){ try{
        // Create file FileWriter
        fstream = new
FileWriter("out.txt",true);
        BufferedWriter out = new
BufferedWriter(fstream); out.write("fail
        to complete the
jump"+"\r\n");
        in.hopCount++;
        out.close();
        }catch (Exception e){//Catch exception if
any
        System.err.println("Error:
" + e.getMessage());
        }
  fail=true;
 }
```

Thus, if the jumps are not yet finished and do not fail, the method would then jump to the next neighbour.

```
if(length>0 &&
!fail){ return
neighbors.get(neighborToAsk).RandomWalk
WithJumps(in,length-1);
}
```

After finishing the jump, the method searches for a specific

file in the file list at the last neighbour in the jump.

```
if(this.fileList != null){
  for(int i=0; i< fileList.size(); i++){
        if(fileList.get(i).FileName.equals(in. file
.FileName) ){                fileFound=true;

try{
        // Create file FileWriter
        fstream = new
FileWriter("out.txt",true);
        BufferedWriter out = new
BufferedWriter(fstream);
        out.write("file is found at
node"+nodeId +"\r\n");
        //Close the output
        stream out.close();
        }catch (Exception e){//Catch exception if
any
        System.err.println("Error:
" + e.getMessage());
        }}}}
        if(fileFound){
              return nodeId;
```

The search will check different cases: it will check whether the file is found, and then will return the current node ID if it is. It will also check whether the hop count is already 3; if it is, it will return –1 (small because the graph does not have huge dimensions). If there are neighbours and the jump does not fail, the search will again return to the function. If the case is not one of these, it returns –100, which means fail, and the number of failures increases by 1.

```
if(fileFound){return nodeId;
                }
else if(in.hopCount >= 3){
        return -1;
        }
else{
        if(neighbors != null && !fail ){
        length = rand.nextInt(3);
try{
        // Create file FileWriter
        fstream = new
FileWriter("out.txt",true);
        BufferedWriter out = new
BufferedWriter(fstream); out.write("RW-
WithJump is Visited node:
"+nodeId+"\r\n");
out.write("RW-WithJump current hopCount:
          "+in.hopCount+"\r\n");
out.write("And the length is: "+length+"\r\n");
        //Close the output stream
        out.close();
        }catch (Exception e){//Catch exception
System.err.println("Error: " + e.getMessage());
        }
        in.hopCount++;
return neighbors.get(neighborToAsk).RandomWalkWithJumps(
in , length);}
        }
return -100;
}//end method Random Walk With Jumps
```

V. Testing and Results

In general, the findings indicate that the value of average hops is approximately 2. In addition, the number of failures is between (1500< #of failures<4500) for 10,000 test times. Surprisingly, the number of failures decreased when the number of nodes increased as the algorithm jumps first before asking for a specific file. The following results are for the number of nodes between 10,000 and 100,000.

TABLE III: ANALYSING RESULTS OF FOUR METHODS FOR NUMBER OF FAILURES

	Flood	RW	RW with NT	RW with Jumps
10000	1153	3733	4489	2220
20000	933	4096	4123	2285
30000	638	3526	3949	1939
40000	583	3581	4018	1248
50000	528	3598	3611	1077
60000	561	2750	3404	1110
70000	419	3560	3669	1127
80000	250	3214	3602	647
90000	295	3400	3231	733
100000	387	3221	2817	857

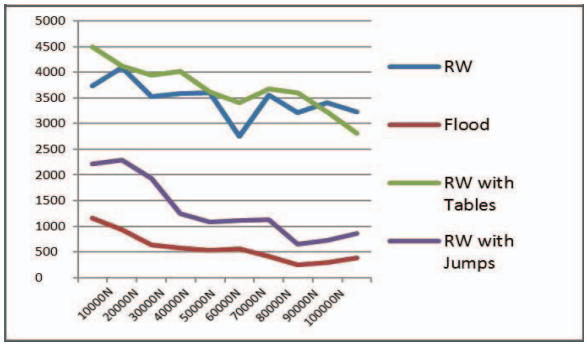

Figure 7: Analysing results of four methods for number of failures

TABLE IV: ANALYSING RESULTS OF FOUR METHODS FOR AVERAGE HOPS

	Flood	RW	RW with NT	RW with Jumps
10000	4.8500	3.9782	3.9151	2.9979
20000	4.8830	4.1479	4.0993	2.991
30000	4.9899	4.4332	4.2190	2.9906
40000	4.9982	4.4126	4.2155	2.988
50000	5.0326	4.4500	4.4166	2.995
60000	5.0109	4.9728	4.6250	2.993
70000	5.0067	4.5010	4.4335	2.993
80000	5.0455	4.7446	4.4353	2.9843
90000	5.0302	4.6148	4.7522	2.9919
100000	5.0468	4.7681	5.0731	2.9917

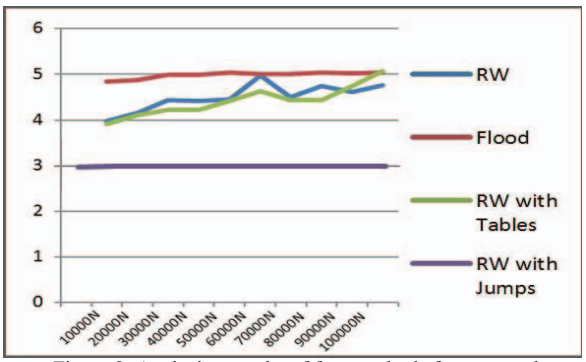

Figure 8: Analysing results of four methods for average hops

As can be seen above, the Random Walk with Jumps algorithm has the best score in average hops because it jumps first then asks for a specific file. The number of failures is not as minimised as in Flooding, but is better than the Random Walk and Random Walk with Neighbours Table. Thus, this method beats the best values in the three methods, which are the number of failures as Flooding and average hop as Random Walk and Random Walk with Neighbours Table.

VI. Conclusion and Future work

Social life, which has promoted the success of the Gnutella network, might change, causing the network to fade. However, P2P is recognised as one of those rare things that simply is too good to go away.

The open architecture, achieved scale, and self-organising structure of the Gnutella network make it an interesting P2P architecture for examination [9]. The measurement and analysis techniques used here also can be used for most P2P systems to enhance general understanding of the design.

This paper analysed the Gnutella network in an effort to study its current methods. The study then compared three of the methods, namely Random Walk, Flooding and Random Walk with Neighbours Table, in an effort to identify which has the least average hops and/or number of failures, and which has the highest. Subsequently, the paper proposed a new query search method, known as Random Walk with Jumps. The new method has proven to be a much better approach with a small number of failures and a minimum number of hops.

There are two potential directions for future study. First, if the average time the researchers have already measured for each of the three methods had been recorded, the achieved results would prove that the new method also has the least average time—not only the least average hops. A second direction is centred on improving the new method to achieve a fewer number of failures.

VII. References

[1] A. Basu, S. Fleming, J. Stanier, S. Naicken, I. Wakeman, V.Gurbani, ―The state of peer-to-peer network simulators‖. *ACM Computing Surveys (CSUR)*, 45(4), 2013.

[2] Y. Wang, X. Yun, Y. Li, "Analyzing the Characteristics of Gnutella Overlays," *Information Technology,* pp.1095-1100, IEEE, 2007.

[3] Google code (2010, May.). ―Gnutella-peersim Experiment with gnutella‖. [online] available: https://code.google.com/p/gnutella-peersim/ (Access Date: 12 May, 2015)

[4] S. Tanenbaum Andrew, J. W. David, ―Computer Networks‖, *Pearson Publisher,* 5th Edition, ISBN-10: 9332518742, 2010.

[5] R. Beraldi, ―Random walk with long jumps for wireless ad hoc networks, ‖ *Department of Computer and Systems Engineering, University of Rome,* Italy, vol. 32, pp294–306, 2009.

[6] F.Hermann , ―Scalability of the Gnutella Network and Business Opportunities of Peer-to-Peer Networking‖, *GRIN Verlag Publisher,* ISBN 3638136329 , 2002.

[7] J. Harris 2005, ―A Scalable & Extensible Peer-to-Peer Network Simulator‖, *Ottawa-Carleton Institute for Computer Science*, Canada, April 2005.

[8] M. Baker and R. Lakhoo, ―Peer-to-peer simulators‖, *ACET University of Reading*, UK, May 2007.

[9] M. Ripeanu, ―Peer-to-Peer Architecture Case Study: Gnutella Network‖, *The University of Chicago*, USA, 2001.

[10] P. Murthy, ―GTKgREP - Design and Implementation of a Gnutella-based Reputation Management System‖, *North Carolina State University*, USA, 2003.

[11] V. Aggarwal, A. Feldmann, S. Mohr, _Implementation of a P2P system within a network simulation framework‘, *Technische Universität München*, Germany, 2005.

[12] S.K. Dhurandher, S. Misra, M.S. Obaidat, I. Singh, R. Agarwal, B. Bhambhani", Simulating Peer-to-Peer networks", pp.336,341 , *IEEE/ACS,* 2009.

[13] G. Yutang, L. Wanli, L. Bin, "Improved Resource Discovery Algorithm on Gnutella Based on P2P Networks," *Control Conference,* pp.599, *IEEE,* 2007.

[14] T. Lin, H. Wang, ―Search Performance Analysis in Peer-to-Peer Networks‖, *National Taiwan University*, Taipei, Taiwan , 2003.

[15] M. Castro, M. Costa and A. Rowstron, ―Should we build Gnutella on a structured overlay? ‖, *Microsoft Research*, Cambridge, UK, 2003.

[16] A. Prakash, ―A Survey of Advanced Search in P2P Networks‖, *Department of Computer Science, Kent State University,* 2006.

[17] Keqin Li, "Performance analysis and evaluation of random walk algorithms on wireless networks," *Parallel & Distributed Processing, Workshops and Phd Forum*, pp.1-8, *IEEE,* 2010.

[18] J. Cui, J. Guo, C. Zhang ,X. Chang, ―Implementation of random walk algorithm by parallel computing ‖, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2477 – 2481, *IEEE,* 2012.

[19] D. Stutzbach, R. Rejaie. N. Duffield, S. Sen, W. Willinger, ―On unbiased sampling for unstructured peer-to-peer networks‖, IEEE/ACM Transactions on Networking (TON) Journa, 17(2), pp.377-390 . April2009, *IEEE/ACM.*

[20] Gkantsidis, Christos, Milena Mihail, and Amin Saberi. "Random walks in peer-to-peer networks." *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. *IEEE,* 2004.

[21] N. Alon, C. A. Vin, M. Koucky, G. Kozma , Z. Lotker and M. Tuttle, ―Many Random Walks Are Faster Than One‖. *Combinatorics, Probability and Computing, 20*, pp. 481-502.