

April 2015

EFFICIENT SEARCH ALGORITHM DESIGN FOR UNSTRUCTURED PEER-TO-PEER NETWORKS

M.LAKSHMI SURESH

St.Anns College of Engineering & Technology, Chirala, AP, India, suresh.mandlem@gmail.com

M.LAKSHMI BHAI

St.Anns College of Engineering & Technology, Chirala, AP, India, lakshmibaimaddala@yahoo.com

S AMARNATH BABU

St.Anns College of Engineering & Technology, Chirala, AP, India, amardots@yahoo.co.in

Follow this and additional works at: <https://www.interscience.in/ijcct>

Recommended Citation

SURESH, M.LAKSHMI; BHAI, M.LAKSHMI; and BABU, S AMARNATH (2015) "EFFICIENT SEARCH ALGORITHM DESIGN FOR UNSTRUCTURED PEER-TO-PEER NETWORKS," *International Journal of Computer and Communication Technology*. Vol. 6 : Iss. 2 , Article 16.

Available at: <https://www.interscience.in/ijcct/vol6/iss2/16>

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

EFFICIENT SEARCH ALGORITHM DESIGN FOR UNSTRUCTURED PEER-TO-PEER NETWORKS

¹M.LAKSHMI SURESH, ²M.LAKSHMI BHAI & ³S AMARNATH BABU

St.Anns College of Engineering & Technology, Chirala, AP, India
Email:suresh.mandlem@gmail.com, lakshimbaimaddala@yahoo.com, amardots@yahoo.co.in

Abstract: Peer-to-peer systems are becoming increasingly popular, with millions of simultaneous users and a wide range of applications. Understanding existing systems and devising new peer-to-peer techniques relies on access to representative models derived from empirical observations. Due to the large and dynamic nature of these systems, directly capturing global behavior is often impractical. Sampling is a natural approach for learning about these systems, and most previous studies rely on it to collect data. This paper addresses the common problem of selecting representative samples of peer properties such as peer degree, link bandwidth, or the number of files shared. A good sampling technique will select any of the peers present with equal probability. However, common sampling techniques introduce bias in two ways. First, the dynamic nature of peers can bias results towards short-lived peers, much as naively sampling flows in a router can lead to bias towards short-lived flows. Second, the heterogeneous overlay topology can lead to bias towards high-degree peers. We present preliminary evidence suggesting that applying a degree-correction method to random walk-based peer selection leads to unbiased sampling, at the expense of a loss of efficiency.

Keywords: Peer-to-peer, performance analysis, search algorithm

1. INTRODUCTION

Peer-to-peer (P2P) systems are becoming increasingly popular, with millions of simultaneous users [1] and covering a wide range of applications, from file-sharing programs like LimeWire and eMule to Internet telephony services such as Skype. Understanding existing systems and devising new P2P techniques relies on having access to representative models derived from empirical observations of existing systems. However, due to the large and dynamic nature of P2P systems, it is often difficult or impossible to directly capture global behavior. Sampling is a natural approach for learning about these systems using light-weight data collection, relied on by most previous studies (e.g., [4]). One challenge, however, is ensuring that the samples are representative (or unbiased).

This paper addresses the common problem of selecting representative samples of peer properties such as peer degree, link bandwidth, or the number of files shared [4]. To examine peer properties, any sampling technique needs to locate a set of peers in the overlay and gather data from them. Initially, the sampling program is aware of a handful of peers and leveraging them to learn about additional peers. Typically, the sampling program queries known peers to learn about their neighbors, incrementally exploring a fraction of the overlay graph. A good sampling technique will select any of the peers present with equal probability. However, as we will show, commonly used sampling techniques can easily introduce significant bias in two ways. The first cause of bias is the highly dynamic nature of these systems. It is easy to imagine the overlay as a static graph from

which we want to collect a set of peers. However, gathering a set of samples takes time, and during that time the graph will change. In Section II-A, we show how this often leads to bias towards short-lived peers and explain how to overcome this difficulty. The second significant cause of bias is the graph properties of the P2P topology. A naive approach will be heavily biased towards high-degree peers. As the sampling program explores the graph, each link it traverses is much more likely to lead to a high-degree peer than a low-degree peer. We describe different techniques for traversing the overlay to select peers in Section II-B and evaluate them in Section III via simulation. In this preliminary work, we simulate using two types of graphs: ordinary random graphs and an actual snapshot of the Gnutella graph topology [22]. In our ongoing work, we are adding other types of random graphs, such as certain power-law random graphs and small-world graphs, to explore the robustness of the considered techniques to different types of graph structures. By comparing and contrasting the performance of different techniques in different settings, we can gain a better understanding of the most efficient techniques to consistently yield unbiased (or only slightly biased) samples.

In summary, bias in sampling from P2P systems can be introduced along two axes: (i) temporal (due to differences in peer lifetimes) and (ii) topological (due to differences in peer degree). Our findings show that these factors cause heavy bias in commonly used techniques such as breadth-first search and random walks. We present preliminary evidence suggesting that applying a degree-correction method to random walk leads to unbiased sampling, at the expense of a loss of efficiency. Section IV discusses related work,

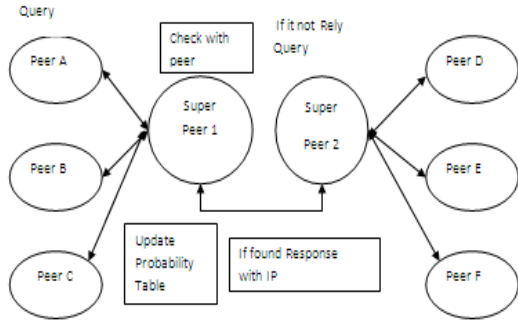


Figure 2: Data Flow Diagram

5. SYSTEM TESTING

In a software development project, errors can be injected at any stage during development. The development of software involves a series of production activities where opportunities for injection of human fallibility's are enormous. Because of human inability to perform and communicate with perfection, software development is accomplished by a quality assurance activity.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for the software engineer. The engineer creates a series of test cases that are intended to demolish the software engineer process that could be viewed as destructive rather than constructive.

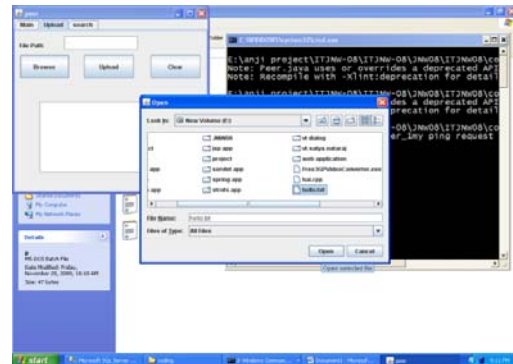
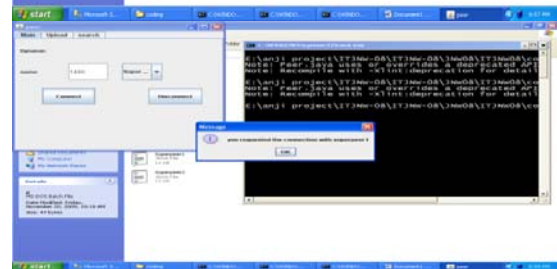
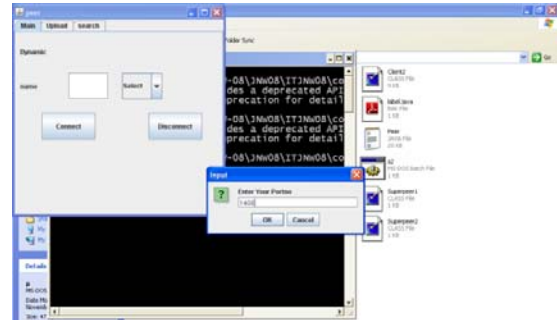
Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for the software engineer. The engineer creates a series of test cases that are intended to demolish the software engineer process that could be viewed as destructive rather than constructive. Equivalence partitioning is a black-box testing method that derives the input domain of program into classes of data from which test cases can be derived. An ideal test case single handedly uncovers a class of errors (e.g. incorrect processing of all incorrect data) that might otherwise require many cases to be executed before the general errors are observed. Equivalence partitioning strives to define a test that uncovers the class of errors, thereby reducing the total number of test cases that uncovers classes of errors, thereby reducing the total number of test cases that must be developed.

Testing case testing for equivalence partitioning is based on an evaluation of equivalence class for an input condition if a set of objects can be linked by relationship that are symmetric, transitive and reflexive, an equivalence is present. An equivalence class represents a set of valid or invalid states for input condition. Typically an input condition is a specific numeric value, a range of values, a set of related values, or a condition.

Code Testing: The code test has been conducted to test the logic of the program. Here, we have tested with all possible combinations of data to find out logical errors. The code testing is done thoroughly with all possible data available with library. Program Testing: Program testing is also called unit testing.

The modules in the system are integrated to perform the specific function. The modules have been tested independently, later Assembled and tested thoroughly for integration between different modules. System Testing: System testing has been conducted to test the integration of each module in the system .It is used to find discrepancies between the system and its original objective. It is found that there is an agreement between current specifications and system documentation. Software Testing is carried out in three steps

The first step includes unit testing where in each module is tested to provide his correctness, validity and also determine any missing operations. Errors are noted down and corrected immediately. Unit testing is the import and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.



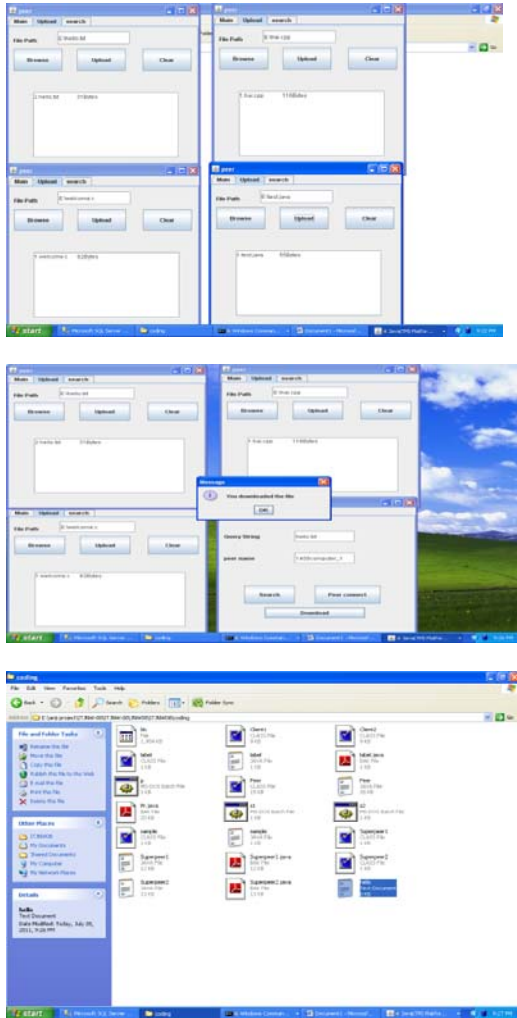


Figure 3: Screen shots

6. CONCLUSION

In this paper we have explored several techniques for sampling from P2P systems. One of our contributions is to show that unbiased sampling must allow the same peer to be selected multiple times to avoid bias correlated with peer sessions lengths.

We simulated each technique over ordinary random graphs as well as a real Gnutella topology and evaluated how much bias and correlation they introduce as well as their efficiency. We found that the commonly used BFS technique, while efficient, introduces significant sampling bias. Conducting random walks is also significantly biased and additionally is inefficient. The random stroll technique corrects the inefficiency, but remains significantly biased. Each of these techniques are biased due to the influence of the degree distribution. We describe a “degree correction” modification to the random walk and random stroll techniques that corrects the bias, resulting in samples that appear just as accurate as using an oracle. However, there is a significant decrease in efficiency when using these

techniques. In our ongoing work, we are extending our study to include additional types of random graphs, such as power-law random graphs and small-world graphs. By comparing and contrasting the performance of different techniques in different settings, we can gain a better understanding of the most efficient techniques to yield unbiased samples. Additionally, we are exploring techniques for estimating global properties, such as the number of peers in a P2P system or the diameter of an overlay network by exploring only a fraction of the graph.

REFERENCES:

1. 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. 2006
2. D. Milojevic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, “Peer-to-Peer Computing,” Technical Report HPL-2002-57, HP, 2002.
3. B. Yang and H. Garcia-Molina, “Improving Search in Peer-to-Peer Networks,” Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02), pp. 5-14, July 2002.
4. C. Gkantsidis, M. Mihail, and A. Saberi, “Random Walks in Peer-to-Peer Networks,” Proc. IEEE INFOCOM '04, pp. 120-130, 2004.
5. C. Gkantsidis, M. Mihail, and A. Saberi, “Hybrid Search Schemes for Unstructured Peer-to-Peer Networks,” Proc. IEEE INFOCOM '05, pp. 1526-1537, 2005.
6. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and Replication in Unstructured Peer-to-Peer Networks,” Proc. 16th Ann. Int'l Conf. Supercomputing (ICS '02), pp. 84-95, June 2002.
7. Z. Ge, D.R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, “Modeling Peer-Peer File Sharing Systems,” Proc. IEEE INFOCOM '03, pp. 2188-2198, 2003.
8. S. Saroiu, P.K. Gummadi, and S.D. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”. MMCN, Jan. 2002.
9. S. Jiang, L. Guo, X. Zhang, and H. Wang, “LightFlood: Minimizing Redundant Messages and Maximizing Scope of Peer-to-Peer Search,” IEEE Trans. Parallel and Distributed Systems, vol. 19, no. 5, pp. 601-614, May 2008.
10. D. Tsoumakos and N. Roussopoulos, “Adaptive Probabilistic Search for Peer-to-Peer Networks,” Proc. Third Int'l Conf. Peer-to-Peer Computing (P2P '03), pp. 102-109, Sept. 2003.
11. B. Yang and H. Garcia-Molina, “Improving Search in Peer-to-Peer Networks,” Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02), pp. 5-14, July 2002.

