

Reducing Maintenance Overhead in DHT Based Peer-to-Peer Algorithms

Zhiyong Xu, Rui Min and Yiming Hu

Department of Electrical & Computer Engineering and Computer Science

University of Cincinnati

E-mail: {zxu,yhu}@ececs.uc.edu

Abstract

DHT based Peer-to-Peer (P2P) algorithms are very promising for their efficient routing performance. However, most commercial P2P systems do not adapt DHT algorithms and still use central facilities or broadcasting based routing mechanisms. One reason impeding the DHT algorithm popularity is the routing information maintenance overhead in DHT algorithms, it generates considerable network traffic and increases P2P system complexity, especially in a highly dynamic environment. In this paper, we discuss its effects on DHT routing performance and propose our solution to reduce this overhead.

1 Introduction

A lot of research papers discussed the Distributed Hash Table (DHT) based P2P routing algorithms (Chord [1], Pastry [2], Tapestry [3] and CAN [4]). Although using DHT based algorithms is a very promising approach, a lot of difficult problems, such as routing information maintenance, storage management, fault tolerance and security, impede their popularity on Internet. Until now, DHT algorithms are not widely used in commercial systems yet, most P2P file sharing systems are still using non-structured P2P mechanisms.

A big issue in current DHT based P2P algorithms is the high overhead of maintaining DHT routing data structures. When a node joins/leaves the system, the affected data structure on some existing nodes must be updated accordingly to reflect the change. For nodes which join/leave the system frequently, the system will generate a lot of routing information update traffic. It is not only increase the bandwidth consumption, but also affect the efficiency of DHT based routing algorithms.

The rest of the paper is organized as follows. In Section 2, we briefly analyze the maintenance overhead in DHT algorithms. We present our solution in Section 3. The related work is discussed in Section 4. Finally, in Section 5, we conclude our proposal and give the future work.

2 DHT Maintenance Overhead

In DHT based P2P systems, a routing procedure can be finished fast and efficiently. To achieve this purpose, each node must

maintain certain amount of system routing information. As a node joins/leaves the system, the routing data structure must be created on this node, the affected routing information on several other nodes must be modified accordingly. To make the system work properly, each node periodically send "hello" message to its neighbors to check their availability. Clearly, DHT algorithms achieve better routing performance than non-structured P2P algorithms at the cost of increasing system routing information maintenance overhead.

In a large-scale P2P system, this overhead can be high, especially in a dynamic environment, system generates considerable routing maintenance traffic. When system detects a node joins, it will send update messages to all affected nodes. However, for a frequently join/leave node, after the affected nodes modified their routing information, it may leaves the system and the system need to send node leave messages to all these nodes to update their routing information again which causes a routing information thrashing problem. Since substantial users use unstable dial-up connections [5], this is a big problem which decreases DHT system routing performance.

3 Proposed Solution

Current DHT system design did not pay enough attention on the diversity of system nodes. Although the principle of P2P system is the equal treatment of all system members, peers have different behavior and have different effect on system performance. Some peers are powerful and stable which can make more contributions than other peers, some other peers only increase maintenance overhead without contribute much.

As we mentioned, in DHT algorithms, when a node joins the system, the corresponding routing information must be created on it and affected routing tables on other nodes must be modified. As a node leaves the system or a node failure occurs, the affected routing information on some nodes must be updated. For heavily dynamic nodes, system maintenance work is useless. For relative stable nodes, system has no much maintenance work to do. Our purpose is to use stable nodes to take most system workload and reduce the DHT algorithm maintenance overhead.

We evaluate the availability of each node during the time it joins the system. We prefer to add stable nodes into routing data structures instead of frequently join/leave nodes. A simple

scheme is used to measure a node's availability. An *average live time table* **ALT** is used to maintain the duration a node stay in the system. An ALT table has 10 entries which records the durations for the most recent 10 appearances. If the table has 10 records already when it joins the system, the value in the oldest entry will be evicted from the table. An *Average Duration Time* **T** is used to record the average value. To address sudden node failure problem, the node updates the present ALT entry every 1 or 2 minutes. Thus we can have a proximate accurate value even in case of a node failure. Another method is to let one of its neighbor nodes to perform this task. Since neighbor nodes must change "hello" message periodically among them, this work does not bring extra overhead.

In our proposal, each entry in routing tables also records **T** for that node. In case a new node joins the system, before an affected node modify its routing table, it compares **T** value of this newly added node and the node in the affected entry. If the new node has a higher **T**, the entry will be updated, otherwise, nothing is done. For nodes which join the system for the first time, its **T** is set to 0 and it is unlikely to be added in routing tables.

Another strategy to reduce the maintenance overhead is delayed update. In our system, the nodes in current routing tables are relative stable. Because a large portion of routing failures are caused by temporal problems such as network congestion, many of them can be recovered in a short time. In original DHT design, during a routing procedure, if the current node can not get the response from the node of the next routing hop, it notices that node is fail and the affected entry in its routing table should be replaced by another node. In our solution, we assume most of these problems are not the real node failure. The system checks **T** of the node in this entry, if it is large than a certain threshold (e.g. 30 minutes), we only mark this entry as *possible failure* and do not replace it. If a new request comes next, this entry still can be chosen. Only there's a certain number of failure occurs, we believe this node is fail and begin the replacement procedure.

Another situation that an entry in a routing table is updated happens when a node ensure one of its neighbor node is fail. Since neighbor nodes exchange "hello" messages periodically, a node *N* can detect a failure of its neighbor within a short time. If the failed node *F* has a high **T**, which means *F* is relative stable, node *N* will broadcast this message immediately and the affected routing tables are updated. If node *F* has a low **T**, *N* will check it again several times, only in case a certain time (e.g. 30 minutes) passed, *N* will broadcast the node failure message. Since the original DHT design do a lot of useless maintenance work for frequently join/leave nodes, our strategy can effectively reduce this overhead.

4 Related Works

Compare to the non-structured P2P systems [6, 7], DHT algorithms can greatly improve system routing performance at the cost of routing information maintenance. All the DHT algorithms have a well-designed data structure and routing algorithm to achieve this goal. However, DHT algorithms are still in the preliminary stage, a lot of issues still need to be addressed. Current DHT researches are still concentrated on improving the

routing performance, the DHT data structure maintenance overhead and its effects are not well considered and evaluated. In this paper, we analyze this problem and propose our solution.

Using supernodes in P2P systems is a hot research topic recently [8, 9, 10, 11]. In these systems, the powerful nodes are selected as supernodes and take most system tasks. By creating another supernode overlay network on existing P2P network, the routing tasks can finished much faster. Our approach can be viewed as the similar strategy, the supernodes are generally stay longer time than other nodes. However, these systems do not address the DHT maintenance problem, the routing information still need to be updated each time a node join/leave the system. Our approach can reduce the overhead by preventing frequently join/leave nodes to pollute system routing information.

5 Conclusions and Future Work

In this paper, we discuss the routing data structure maintenance problem in DHT based P2P algorithm and propose the solution. Our work is still in early stage, in the future, we will perform quantitative analysis on DHT routing data structure maintenance overhead, design an optimal node availability measurement mechanism and routing table maintenance strategy, and we will also conduct trace-driven simulations to evaluate the efficiency of our solution.

References

- [1] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." TR-819, MIT., Mar. 2001.
- [2] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware, Heidelberg, Germany*, pp. 329–350, Nov. 2001.
- [3] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant widearea location and routing." TR UCB/CSD-01-1141, U.C.Berkeley, CA, 2001.
- [4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network." Technical Report, TR-00-010, U.C.Berkeley, CA, 2000.
- [5] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in (*MMCN '02*), (San Jose, CA, USA), January 2002.
- [6] Napster, "http://www.napster.com."
- [7] Gnutella, "http://www.gnutella.wego.com."
- [8] KaZaA, "http://www.kazaa.com/."
- [9] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade:landmark routing on overlay networks," in *IPTPS, Cambridge, MA*, March 2002.
- [10] Z. Xu and Y. Hu, "SBARC: A Supernode Based Peer-to-Peer File Sharing System," in (*ISCC, Kemer-Antalya, Turkey*), June 2003.
- [11] B. Yang and H. Garcia-Molina, "Designing a Super-peer Network," in (*ICDE, Bangalore, India*), March 2003.