A Chord-Based Novel Mobile Peer-to-Peer File Sharing Protocol

Min Li[1], Enhong Chen[1], and Phillip C-y Sheu[2]

[1] Department of Computer Science and Technology,
University of Science and Technology of China,
Hefei, Anhui, 230027, P.R. China
minli@mail.ustc.edu.cn, cheneh@ustc.edu.cn
[2] Department of EECS,
University of California,
Irvine, CA 92697
psheu@uci.edu

Abstract. With the increasingly developed technology of mobile devices and wireless networks, more and more users share resources by mobile devices via wireless networks. Compared to traditional C/S architecture, P2P network is more appropriate for mobile computing environment. However, all existing P2P protocols have not well considered the characteristics and constraints of mobile devices and wireless networks. In this paper, we will present a novel mobile P2P protocol, M-Chord, by adopting hierarchical structure and registering mechanism on the basis of Chord. The experimental results show that M-Chord system has high-efficiency and good robustness in mobile P2P network.

1 Introduction

With the increasing technology of mobile devices and wireless networks, it gets more and more prevalent to share various resources by mobile devices such as a PDA or a mobile phone via different types of wireless access networks such as GPRS and IEEE 802.11 wireless LAN. On the other hand, compared to the constraints of traditional Client/Server (C/S) architecture, such as the systemic fragility, the bottleneck of system performance caused by high-load of the center server, the overmuch bandwidth consumption for broadcasting message and so on, Peer-to-Peer (P2P) network is more appropriate to employ in mobile computing environment because it adopts distributing services among equal nodes and improves the scalability and reliability of the whole system. However unfortunately, all existing P2P protocols have not specially considered the problems of wireless joining for mobile devices, for instance, limited CPU and memory of mobile devices, intermittent disconnection, limited bandwidth, high transmission delay etc. So how to obtain more efficient and effective mobile P2P techniques have recently become prominent discussion topics.

For the new generation of scalable P2P systems that support distributed hash table (DHT) functionality, such as Tapestry [1], Pastry [2], Can [3] and Chord [4], files are associated with a key by hashing its title or its content, and each node is responsible for storing a certain range of keys. Each DHT system employs a different routing algorithm and has its own attractive respects and disadvantages at the same time. But we appreciate the simplicity, scalability and high-efficiency of Chord much more. Although some researchers have presented a mobile P2P protocol M-CAN [5] based on CAN, the theory of hierarchical structure and registering mechanism can not be applied to Chord directly. In this paper, we will propose a novel mobile P2P file sharing protocol, M-Chord, by ameliorating Chord protocol and adopting hierarchical structure and registering mechanism aim to accord with the characteristics of MP2P.

2 Design and Implementation of M-Chord

As an extensible P2P routing algorithm, Chord adopts the simple logic structure, simple systematic interface and uses one-dimensional circular key space. Considering the constraints of mobile environment, we have modified and improved the base Chord protocol to obtain an efficient lookup protocol in MP2P, M-Chord.

2.1 Hierarchical Structure of M-Chord

M-Chord adopts hierarchical structure to organize mobile peers and it introduces the theory of register mechanism used in M-CAN to manage resources. There are two kinds of nodes in M-Chord, super nodes and ordinary nodes. Super nodes are the nodes having larger memory, better computing capability and more reliable connection. Ordinary nodes are associated with super nodes by the register mechanism. Firstly, when a sharing file is published in M-Chord system, it will be assigned a file ID according to its content and title by hash function. Every super node manages a range of file IDs separately. The node which has sharing files will be registered on some super nodes according to the IDs of its shared files. The node which has no sharing files will be appointed a super node with minimum load. A super node will be registered on itself but it should also present the information of all its sharing files to its corresponding super nodes. Every ordinary node needs to record the IDs and addresses of its super node(s). Every super node needs to maintain two tables: one is the routing finger table, the other is sharing files directory, which records the information about its registered files, such as the file IDs, the registered nodes' addresses etc. For super nodes, we use Chord to manage them.

2.2 Routing in M-Chord

Before a source node sends out its file access request, it must calculate the ID of its wanted file firstly. Then the source node would submit a request containing this ID to the source super node which the source node is registered on [5]. After receiving the request, the source super node would lookup its finger table and go on with the routing process until the destination super node whose ID space covers the ID of the wanted file receives the request. The routing process on the ring is the same as Chord. Then the destination super node would launch a lookup process locally. If the destination super node can find the very file ID in its directory, which means the destination node which owns the wanted file is registered locally, it would return the destination node's address to the source node. After the source node gets the destination node's address, it would try to communicate with the destination node directly. A lookup is considered to fail if it returns the wrong node among the current set of participating nodes at the time the sender receives the lookup reply (i.e., the destination node has left or failed already), or if the sender receives no reply within some timeout window.

2.3 Node Join

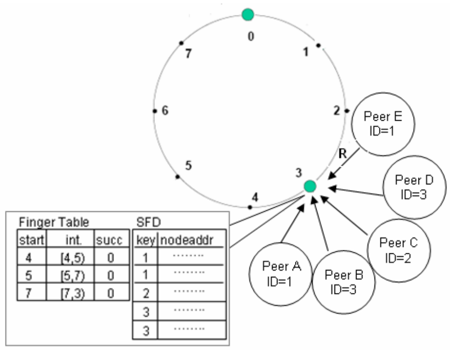
When a new node joins M-Chord, it must register itself on one or more appropriate super nodes according to the IDs of its shared files. If succeed in registration, the corresponding super nodes should update their sharing file directory in time.

In order not to make super nodes be the bottleneck of the whole system, we set a rule that any super node can manage *n* nodes at most. If a super node manages more than *n* nodes, a split process would be triggered. In this paper, we will present two kinds of split strategies, real split and virtual spilt.

Real split strategy. In the "real split" process, the original super node (denoted as OS) will first choose a new super node (denoted as NS) from its registered nodes. NS is the node with the best connection among registered nodes. We assign an ID to the new super node by the following formula**.**

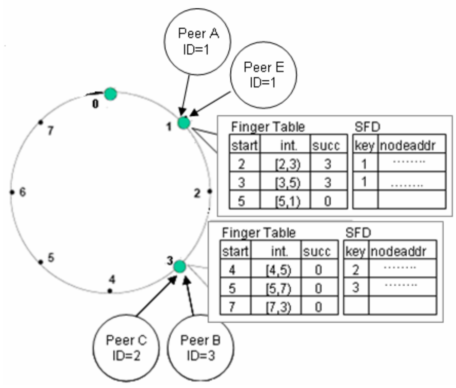
$$NS.id = \lfloor (OS.predecessor.id + OS.id)/2 \rfloor \tag{1}$$

Unlike Chord system, which generate the node identifiers by hashing its IP address to an *m*-bit space, M-Chord system will identify the ordinary node uniquely by its address, and to the super node, it will assign its ID with the median of two existing super nodes to ensure the uniqueness of the IDs of super nodes too.



start	int.	succ
4	[4,5)	0
5	[5,7)	0
7	[7,3)	0

key	nodeaddr
1	....
1	....
2	....
3	....
3	....



start	int.	succ
2	[2,3)	3
3	[3,5)	3
5	[5,1)	0

key	nodeaddr
1	....
1	....

start	int.	succ
4	[4,5)	0
5	[5,7)	0
7	[7,3)	0

key	nodeaddr
2	....
3	....

Fig. 1. Before real splitting of super node 3, n=4 **Fig. 2.** After real splitting of super node 3, n=4

The registered nodes will be divided into two groups. One group continues to be managed by the original super node, and the other group is managed by the new super node. Then we should go on with four steps. (1)Generate the finger table of the new super node *NS*. (2)Update neighbor information of *OS* and *NS*. (3) Update sharing file directories of *OS* and *NS*. (4)Update finger tables of all super nodes except *NS*.

Fig.1 and Fig.2 show an instance for super node splitting. Suppose one super node can manage four ordinary nodes at most. Fig.1 shows the state when peer E joined and registered on super node 3. Super node 3 has exceeded the management maximum, so it will be split. Fig.2 shows the state after super node 3 splitting. A new super node, super node 1 is generated and the rest registered nodes are divided into two groups.

Virtual split strategy. The rationale of virtual split strategy is very simple. Like real split, in virtual split process the original super node *OS* will first choose a new super node *NS* from its registered nodes, then divide the rest registered nodes into two groups. Each group has the same number of nodes and each super node manages one group. Unlike real split strategy, we assign the same ID as *OS* to *NS*. Then *NS* and *OS* will update their sharing file directories to keep consistent. Because there is not a new node ID generated on the M-Chord ring, it is unnecessary to update the finger tables of other super nodes. We only need to copy the finger table and neighbor information from *OS* to *NS*. But we should note that during the process of lookup operation, one position of routing path may relate to more than one super node. If we need to check the finger table of this position, we only need to check one node randomly. Moreover, the destination super node ID may also correspond to more than one super node. In this case we need to check all these virtual super nodes' sharing file directories to lookup the destination node.

In fact, the processes of node joining and super node splitting are also the system construction process. At first, there is only one node that covers the whole ID space in the system. In this case, any node will be registered on the original super node until the number of registered nodes exceeds the limit n. Then the split process will be triggered. If sharing files are distributed equally on the whole ID space, real spilt will balance the load of M-Chord ring perfectly. But if there are too many sharing files within a certain continuous file ID range, real splitting may cause a "local saturation" situation. That is to say, several continuous positions on the ring have existed super nodes and the super node in this series can not split any more although there are many vacant positions on the other part of the ring. To solve this problem, we set the rule that if the same file is shared and registered so many times by different nodes as to exceed a certain limit, it is forbidden when the file attempts to be published by later joined nodes. Compared with virtual split, real split will increase some expenses, such as the expense of updating all super nodes' finger tables. However although virtual split is simple and highly efficient, it will cause the whole system load-unbalance because it allows too many nodes congregating on one position of M-Chord ring. So in practice, we can combine these two strategies to achieve the best performance. In our simulation experiment, we stipulate that to every node ID on the ring, there exist two virtual super nodes at most. If exceeds, real split will be performed.

2.4 Node Departure

When an ordinary node leaves the system, we only need to modify the directory of its super node(s). But if the missing node is a super node, we should extend the file ID space of its successor to cover the *missing* space. If the successor super node manages more than n nodes, a split process would be triggered.

3 Theoretical Analysis and Simulation Results

To evaluate the performance of M-Chord system, we design our simulation based on the platform P2PSIM [6], a P2P simulation software developed by MIT. P2PSIM has provided the interface of Chord and we made some modification and extension on the basal Chord to implement M-Chord. We design two data sets with different *n* standing for the maximum that one super node can manage ordinary nodes with the value 5 and 10 separately. For each data sets, we will compare the average data flow of Chord node, super node of M-Chord and ordinary node of M-Chord, measured by the unit *bw* (bytes/node/s), under different total node number (128, 256, 512, 1024, 2048). Our testing data accords with Kingdata [7]. Fig.3~4 show the result with different parameter value. Clearly we can find that the average data flow of super node in M-Chord is the largest, Chord node is secondary and the minimum is ordinary node of M-Chord. This demonstrates that super nodes have shielded most expenses and the total network bandwidth occupation of the whole M-Chord is reduced remarkably.

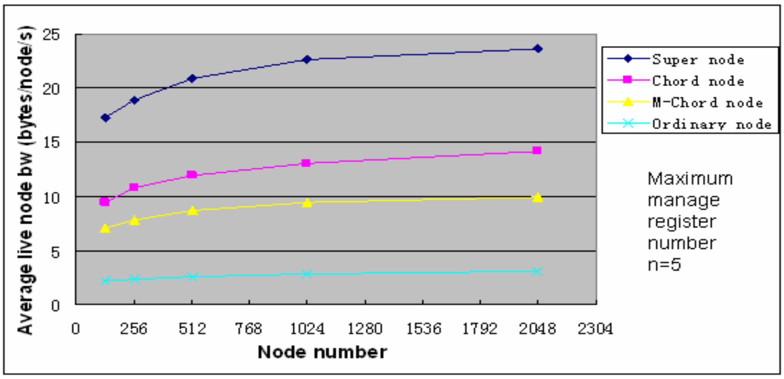


Fig. 3. Node average bandwidth consumption comparison with n=5

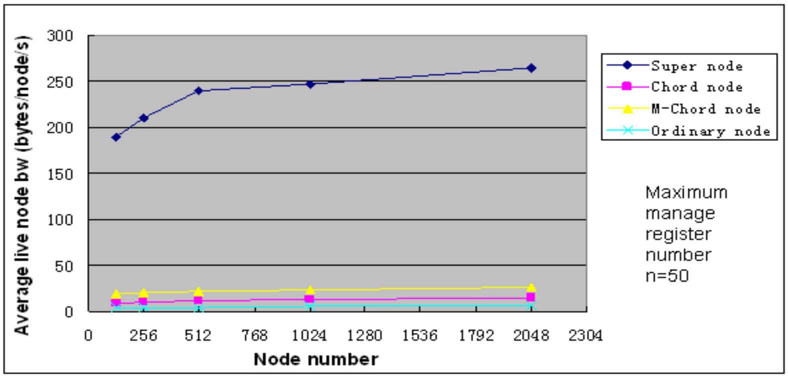


Fig. 4. Node average bandwidth consumption comparison with n = 50

4 Conclusions and Future Work

In this paper, we have presented a mobile P2P protocol M-Chord. The particularities and constraints of MP2P make traditional P2P protocols lowly-efficient and unreliable. Aim to accord with the characteristics of MP2P, we introduce hierarchical structure and registering mechanism into Chord. From our experimental results, we know that compared to Chord, M-Chord greatly reduces the network bandwidth occupation. M-Chord system has high-efficiency and good robustness in mobile computing environment. In the future, we intend to improve the performance of M-Chord by decreasing super node's load and the additional expenses of super node splitting and so on.

Acknowledgements

This work is supported by Natural Science Foundation of China (No.60573077), and the Nature Science Foundation of Anhui Province (No.050420305).

References

1. Ben Y.Zhao, John Kubiatowicz, and Anthony D.Joseph. *Tapestry: An infrastructure for Fault-tolerant Wide-area Location and Routing*. Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Division, 2001.
2. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.
3. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker. A Scalable Content-Addressable Network. Proceedings of ACM SIGGOMM (San Diego, CA, August 2001), pp.161-172.
4. Ion Stoica, Robert Morris, David Karger, M.Frans Kaashoek, Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. Tech. Rep. TR-819, MIT LCS, March 2001.
5. Gang Peng, Shanping Li, Hairong Jin, Tianchi Ma. M-CAN: a Lookup Protocol for Mobile Peer-to-Peer Environment. Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04).
6. http://pdos.csail.mit.edu/p2psim/
7. http://www.cs.washington.edu/homes/gummadi/king/