Back-Up Chord: Chord Ring Recovery Protocol for P2P File Sharing over MANETs*

Hong-Jong Jeong†, Dongkyun Kim†, Jeomki Song†,
Byung-yeub Kim‡, and Jeong-Su Park‡

† Department of Computer Engineering,
Kyungpook National University, Daegu, Korea
{hjjeong, jksong}@monet.knu.ac.kr, dongkyun@knu.ac.kr
† Electronics and Telecommunications Research Institute, Daejoen, Korea
{skylane, pjs}@etri.re.kr

Abstract. Due to a common nature of MANET (Mobile Ad Hoc Networks) and P2P (Peer-to-peer) applications in that they lack a fixed infrastructure, a P2P application can be a killer application over MANET. To save network bandwidth and avoid a point of failure of a directory server, structured P2P systems using DHT (Distributed Hashing Table) like Chord are more suitable for MANET. However, since MANET allows nodes to depart from network, P2P file sharing applications based on Chord lookup protocol should address how to recovery the keys stored at the departed node. In this paper, we propose BU-Chord (Back-Up Chord) in order to detect and recover the departure of nodes by creating and storing a back-up file information in distributed manner. Simulation study proves that our BU-Chord shows off better performance than the existing Chord especially at high departure rate of nodes.

1 Introduction

Recently, research interest in MANET (Mobile Ad Hoc Networks) [1] has increased because of the proliferation of small, inexpensive, portable, mobile personal computing devices. MANET is a wireless network where all nomadic nodes are able to communicate each other through packet forwarding services of intermediate nodes. Besides, from the application's perspective, a P2P (Peer-to-peer) model is prevalent to enable a direct communication between nodes in the network [2]. Many file sharing applications such as Napster [3]and Gnutella [4] rely on this P2P concept. Due to a common nature that MANET and P2P applications both assume a lack of fixed infrastructure, a P2P application can be a killer application over MANET [5]. In centralized systems like Napster, a centralized directory server has the information on who has which files. However, the centralized approach is not suitable for MANET because the server can easily move out of the MANET due to node mobility.

* This work was supported by Electronics and Telecommunications Research Institute (ETRI). The corresponding author is Dongkyun Kim.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 477–484, 2005.

Fully distributed systems like Gnutella do not depend on the existence of a centralized server. A query message for a file search is flooded into the network. Such a distributed approach is also not suitable for MANET because the query flooding produces much traffic in the network with scarce resource. In order to avoid the query flooding, structured P2P systems using the DHT (Distributed Hash Table) mechanism such as Chord [6] were developed. Particularly, Chord distributes files and their references into the network through the DHT technique. Chord forms an overlay network, where each Chord node needs "routing" information about only a few other nodes, as well as "file information" shared among nodes and used in order to know who has requested files.

However, since MANET allows nodes to depart from network, it is difficult to apply the Chord to MANET because it cannot recover the file information. In this paper, we therefore propose BU-Chord (Back-Up Chord) protocol in order to detect and recover the departure of nodes, efficiently. BU-Chord creates and stores a back-up file information in distributed manner. Although this paper applies the BU-Chord to MANET, it can be used in any network environment, where failure of nodes occurs frequently, because the failure of nodes is equivalent to the departure of nodes out of networks.

The rest of this paper is organized as follows. In Section 2, the basic Chord is introduced in short. In Section 3, our BU-Chord protocol is described in detail. We perform the performance evaluation in Section 4, which is followed by concluding remarks in Section 5.

2 Chord: A Scalable P2P Lookup Protocol

Like most of other structured P2P protocols using the DHT technique, Chord defined assignment and lookup mechanisms, where a key is used as a name of a shared file. Each Chord node and key obtain their unique m-bit identifiers by using a base hash function such as SHA-1 [7]. A node's identifier is determined by hashing the node's IP address, while a key identifier is produced by hashing the key. Using the node identifier, the Chord creates an identifier space (from 0 to 2^m - 1), which is called "Chord ring" and distributed in the network.

Chord utilizes a consistent hashing to assign keys to Chord nodes. Key k is assigned to the first node whose identifier is equal to or follows (the identifier of) k in the identifier space. The first node is called the successor of key k, denoted by $successor(k)$. In order for each Chord node to perform a lookup for finding a successor of key k, it should manage its successor, predecessor and finger table. A successor and a predecessor are the next and the previous node in the identifier space, respectively. Finger table consists of m entries. The i^{th} entry in a finger table of node n is $successor(n + 2^{i-1})$, where $1 \leq i \leq m$. In the steady state, in an N-node system, each node maintains information about only O(log N) other nodes, and resolves all lookups through O(log N) messages to other nodes.

Figure 1 shows a Chord ring using m-bit identifier (here, $m = 6$). The ring consists of 10 nodes and has 5 keys assigned. In order for Chord to find a suc-

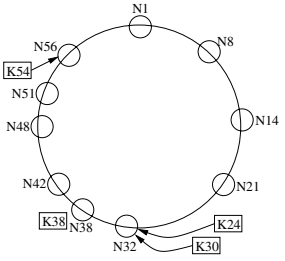


Fig. 1. An example of Chord ring constructed

cessor which will manage a key, it uses a consistent hashing. Therefore, keys are distributed in the Chord ring, which results in providing a degree of natural load balance.

3 BU-Chord: Back-Up Chord

3.1 Motivation

Chord's lookup algorithm enables a wireless network bandwidth to be saved by avoiding the query flooding due to the DHT mechanism. For the purpose of sharing files among Chord nodes, each Chord node should assign a key k (i.e. a name of a shared file) to $successor(k)$ by using the consistent hashing. The searching for a file can be completed by performing a lookup for key k corresponding to the file, just as k is assigned to $successor(k)$. However, since MANET allows nodes to depart from the network at any time, a departure of node causes the network to lose the keys stored at the node. Therefore, any node cannot search for the keys.

To address loss of the keys in the network, other Chord nodes should have the replication of the keys in advance. In this paper, we propose our BU-Chord (Back-Up Chord) to replicate the keys stored at each node into other nodes and recover the keys stored at departed nodes.

In other research work, an effort to improve the reliability of Chord's lookup process over MANET was made [8]. However, for the purpose of developing a P2P file sharing application based on the Chord lookup protocol, the problem should be addressed that the departure of a node causes the keys stored at the node to be lost in the network. Therefore, our BU-Chord can be utilized as their complementary protocol.

3.2 Description of Proposed Approach

In this section, we describe our BU-Chord (Back-Up Chord) protocol to detect a node departure and allow a back-up node to recover the keys stored at the departed node. BU-Chord utilizes a concept of back-up successor and predecessor to replicate the keys stored at a node. Each node performing BU-Chord

protocol is assigned m-bit back-up identifier as well as m-bit identifier by using a hash function such as SHA-1. According to the existing Chord protocol, a node obtains a m-bit identifier by hashing its own IP address. In BU-Chord protocol, each node produces an additional m-bit back-up identifier by hashing its derived m-bit identifier again. A successor of the derived back-up identifier (called back-up successor) is determined by using the same consistent hashing that the existing Chord protocol uses. In BU-Chord protocol, each node requires its back-up successor to replicate its keys and the information on its successor and predecessors of a BU-Chord ring. The back-up successor regards the requesting node as its back-up predecessor and performs a periodical procedure to check if its back-up predecessor is alive in the network through exchange of BEACON/BEACON-ACK message[1].

With the absence of BEACON-ACK, a back-up successor considers that its back-up predecessor departed from the network and performs a recovery process. Since the back-up successor knows who the successor and predecessor of the departed node (say, DN) are, it forces the successor (say, SN) of the DN to update its predecessor with the predecessor of the DN. This procedure is used to just recover a broken Chord ring. As a next step, the keys stored at the DN should be moved to other node. In BU-Chord protocol, since the back-up successor of the DN knows the keys, it can move the keys into the SN, because the SN becomes a successor of the keys. On the other hand, when the back-up predecessor cannot receive its BEACON-ACK from its back-up successor, it is enough to figure out a new back-up successor and replicate its keys and the information on its successor and predecessors of a Chord ring there.

Figure 2 illustrates a recovery operation of our BU-Chord protocol. Assume that N25 has departed and its back-up identifier is B53 and its back-up successor is N57. The predecessor and successor of N25 are N19 and N36, respectively. Through a periodic procedure of N57, N25 is recognized as departed. N57 forces the successor of N25 (here, N36) to update its predecessor with N19 for recovering the Chord ring. Then, N57 moves the information on K22 and K23 into N36.

3.3 Departures of Multiple Nodes

In all cases except a case where both successor of a departed node and its back-up successor disappear at the same time, BU-Chord described in Section 3.2 recovers the keys stored at the departed node. Simultaneous departures of multiple nodes can be broken down into three cases: (i) a departed node (DN) and its successor (SN) move out, (ii) a departed node (DN) and its back-up successor (BN) move out, (iii) a departed node (DN), its successor (SN) and back-up successor (BN) move out.

To address the departures of multiple nodes, some additional mechanism should be executed at back-up successors as described below. The first case can

[1] Proactive MANET routing protocols standardized in IETF (Internet Engineering Task Force) such as OLSR and TBRPF (see [1]) are suitable for reducing the overhead expended to establish a route for exchanging the messages.

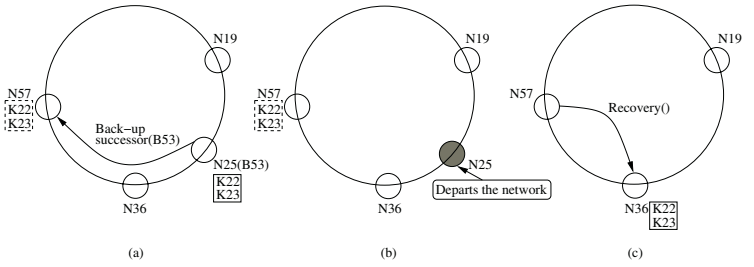


Fig. 2. An Example of BU-Chord Recovery. (a) N57 is a back-up successor of N25. N57 replicates the keys (i.e. K22 and K23) stored at N25. (b) N25 departs the network. (c) N57 moves the keys (i.e. K22 and K23) to N36

be resolved by allowing a back-up successor of an SN to have additional recovery procedure. The BN of a DN attempts to recover the keys into the SN of the DN. The BN checks if the SN works through exchange of BEACON/BEACON-ACK. However, the BEACON/BEACON-ACK exchanging will fail because the SN moved out from the network. Therefore, after the BN considers that both DN and SN moved out, it carries out this following recovery procedure. After BN obtains a back-up identifier of the SN by hashing the node identifier of the SN, it lookups the back-up successor of the SN, i.e. successor(back-up identifier of the SN). Thereafter, the keys stored at the DN are moved into a back-up successor of the SN. Then, the back-up successor of the SN also moves the keys into a successor of the SN.

Figure 3 shows an example of the first case. Suppose that in Figure 2, a successor of N25, i.e. N36 also moved out. The back-up identifier of N36 is B47 and its back-up successor is N49. The predecessor and successor of N36 are N25 and N39, respectively. N57, which is a back-up successor of N25, becomes aware of the departure of N25 and sends a recovery request to N36, the successor of N25. However, the trial will fail because N36 also moved out. Therefore, N57 should try to send a recovery request to a back-up successor of N36, i.e. N49. However, N57 does not know who the back-up successor of the disappeared N36 is. According to our BU-Chord's mechanism through which a back-up successor is determined, the back-up successor of N36 is decided and a recovery request for the keys stored at N25 (i.e. K22 and K23) can be issued. N57 obtains the back-up identifier of N36 (i.e. 47) by hashing the node identifier of N36 (i.e. 36). Therefore, the successor (i.e. N49) of the back-up identifier can be found. Thereafter, N57 provides the back-up successor of N36 (i.e. N49) with the keys stored at N25 (i.e. K22 and K23) and the information on the predecessor of N25 (i.e. N19). The recovery is completed after N49 provides the successor of N36 (i.e. N39) with the keys stored at N25 and the keys stored at N36 (i.e. K35 and K36).

The second case can be resolved by allowing a back-up successor of a BN to perform an additional recovery procedure. After the back-up successor of a BN can recognize that the BN moved out, the keys stored at the BN can be recovered

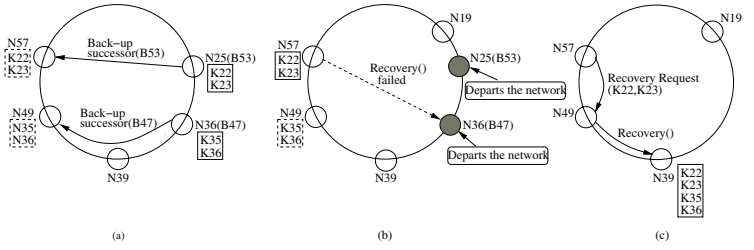


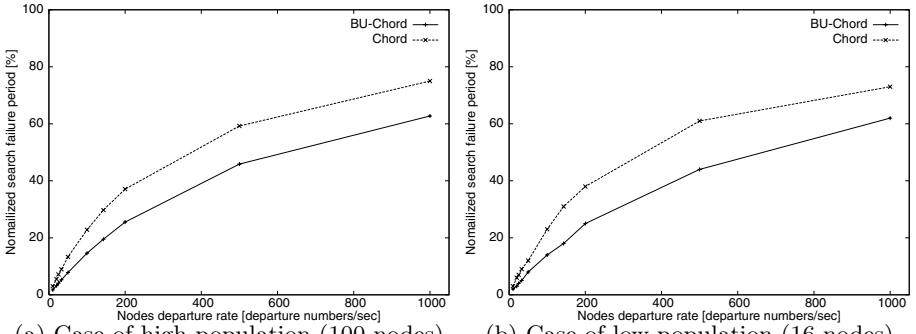
Fig. 3. Both departed node and its successor moved out. (a) N49 and N57 are the back-up successors of N36 and N25, respectively. (b) N25 and N36 depart the network at the same time. (c) N57 provides N49 with K22 and K23. Thereafter, N49 provides N39 with K22, K23, K35 and K36

by moving the keys stored at the BN into a successor of the BN. Thereafter, an additional procedure to check if an actual departure of multiple nodes occurred. The back-up successor of the departed BN executes a recovery of the keys stored at the BN and then, checks if a back-up predecessor of the BN (i.e. the DN) works through exchange of BEACON and BEACON-ACK. Therefore, if two nodes, the BN and its back-up predecessor moved out together, the back-up successor of the BN moves the keys already replicated at the BN, which are originally stored at the DN, into the successor of the DN.

In the final case, the keys stored at nodes, the BN and SN are recoverable through the recovery procedure executed by the BN, SN and each back-up successor of the BN and SN.

4 Performance Evaluation

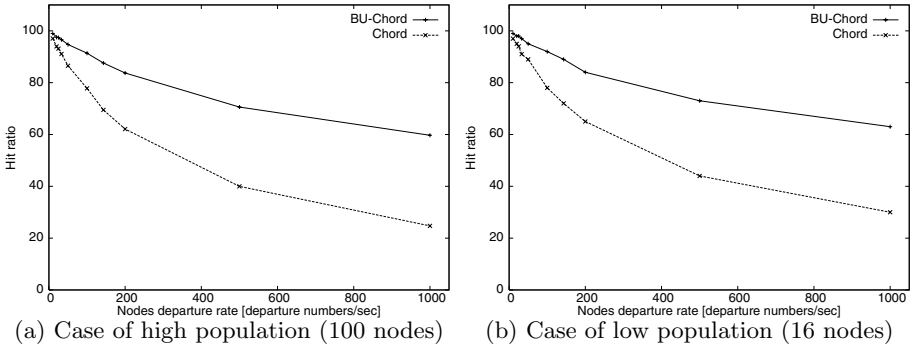
The existing Chord does not define how to recover keys stored at each Chord node. Therefore, we assumed that in Chord, if the keys stored at a departed node are lost in the network, the nodes having the files corresponding to the keys assign them to a new successor of each key. We investigated performances under two kinds of MANETs: (i) high population and (ii) low population. When deploying our BU-Chord over MANET, we assumed that proactive MANET routing protocols are used to reduce the overhead expended to establish a route before moving keys and exchanging BEACON/BEACON-ACK message. For configuring the population of nodes, 100 nodes and 16 nodes are positioned in a grid-style network topology, respectively. We compared BU-Chord with Chord by varying the departure rate of nodes. In our simulation, each node performs search trial periodically by using a normal distribution, where the average interval is 1 second. Each key value which each node would like to find is randomly generated. When a query message succeeds in reaching a successor having the key, it is regarded as search success. Otherwise, we regards the other case as search failure. Therefore, if the keys are not recovered due to node departure, any search for the key will fail.



(a) Case of high population (100 nodes) (b) Case of low population (16 nodes)

Fig. 4. Comparison of normalized search failure period

First, we measured a normalized search failure period (NSFP) according to departure rate of nodes. Since the keys stored at a departed node are recovered by a back-up successor of the departed node, we define NSFP as a portion of the average time during which we cannot perform search success for each key during the total simulation time. As shown in Figure 4, as the departure rate increases, NSFP also increases. Irrespective of departure rate and population of nodes, our BU-Chord shows better NSFP than Chord because BU-Chord allows a back-up successor of a departed node to quickly detect its departure and recover the keys stored at the departed node.



(a) Case of high population (100 nodes) (b) Case of low population (16 nodes)

Fig. 5. Comparison of hit ratio

Second, we investigated hit ratio according to departure rate of nodes, which is defined as a ratio of the number of search success to the total number of search trials. We observed that as the departure rate increases, the hit ratio is decreasing. Obviously, BU-Chord performs better than Chord without regard

to node population and departure rate (see Figure 5). In particular, BU-Chord shows off its performance improvement at high departure rate of node.

5 Conclusion

Using Chord which is a typical structured P2P file sharing application, we can save the network bandwidth by avoiding query flooding in MANET, because keys are distributed in the network. However, when the existing Chord is applied to MANET, departure of nodes forces the keys stored at the nodes to be lost in the network, which result in inability of searching for the keys. However, in our proposed BU-Chord (Back-Up Chord), a back-up successor of each node has the replication of each node's keys and the information on neighboring nodes. Due to the replication technique, the back-up successor detects the departure of node and recovers the keys stored at the departed node. In particular, at high departure rate of nodes in MANET, BU-Chord showed off better performance than Chord irrespective of population of nodes. Although the BU-Chord is applied to MANET in this paper, it can be used in any network environment, where failure of nodes occurs frequently, because the failure of nodes is equivalent to the departure of nodes out of networks.

References

1. Internet Engineering Task Force, "Manet working group charter," http://www.ietf.org/html.charters/manet-charter.html.
2. Gang Ding and Bharat Bhargava, "Peer-to-peer File-sharing over Mobile Ad hoc Networks," IEEE PERCOMW 2004, Mar.2004.
3. Napster, http://www.napster.com.
4. The Gnutella Protocol Specification v0.4.
5. L.B.e. oliveira, I.G. Siqueira and A.A..F Loureiro, "Evaluation of Ad-Hoc Routing Protocol under a Peer-to-Peer Application," IEEE WCNC 2003, Mar. 2003.
6. I. Stoica, R. Morris, D.L. Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networking, Vol. 11, No. 1, Feb. 2003.
7. "Secure Hash Standard," U.S. Dept. Commerce/NIST, National Technical Information Service, Springfield, VA, FIPS 1801-1, Apr. 1995.
8. S.Y. Lee, L. Quan, K.G. Lee, S.Y. Lee, and J.W. Jang, "Trade-off between Message Overhead and Reliability for Peer-to-Peer Search over Mobile Ad Hoc Networks," ISPC COMM 2004, Aug., 2004.