A Peer-to-Peer Search Scheme over Mobile Ad hoc Networks

Sei-yon Lee°, Lan Quan, Kyung-Geun Lee, Tae-kyoung Cho*, Ju-wook Jang°
Sogang University°, Sejong University, SK Telecom*

Abstract – **Chord[6] is an efficient P2P search scheme which uses O(log N) messages per query while Gnutella needs O(N) messages. However, the query hit ratio greatly suffers when the Chord is applied to a mobile Ad hoc network (MANET). In our simulation using NS-2, the hit ratio is as low as 30% under such a mild mobility as walking(2 m/sec). We propose two schemes, Backtracking Chord and Redundant Chord, to improve the query hit ratio over MANET under harsh mobile environment. In Backtracking Chord, query is sent to the preceding finger whenever the query to the current finger timed out due to its absence. The finger is a node which holds the pointer (or pointer to the pointer) to the item to be searched[6]. In Backtracking Chord, query hit ratio and the search time increases as the number of backtracking($0<t<\log N$) increases. Our simulation shows the algorithm achieves up to 88% hit ratio with the mobility of 2 m/sec when $t>4$. In Redundant Chord, r copies of the query are sent to r adjacent fingers simultaneously. Query hit ratio and the bandwidth usage increases as the number of copies(($0<r<\log N$) increases. Simulation under the same mobility shows hit ratio as high as 82% with $r=6$. We also analyze the relationship between the hit ratio, the search latency and the number of query messages(or the bandwidth usage).**

1. Introduction

P2P(Peer-to-Peer) file sharing network which is started with Napster[1] are going on actively with wired internet recently. However, the search in P2P mobile environment has not been developed yet. In general, existing P2P search algorithms in MANET(Mobile Ad-hoc Network) are flooding-based searches, such as Gnutella[2]. However, the flooding-based search algorithm produces too much traffic. For example, in Gnutella, the overhead traffic for maintaining network connectivity occupies more than 50% of the total traffic[3]. Since MANET offers extremely limited bandwidth and also MANET topology changes more dynamically than wired internet environment due to its mobility, the flooding-based search algorithm is not suitable for Mobile Ad-hoc Network[4]. Chord[5], which is one of the well-known decentralized P2P search algorithms, improves the scalability by avoiding the requirement that every node knows about each other node. In an N-node network, each node maintains information only about $O(\log N)$ other nodes, and a lookup requires $O(\log N)$ messages[6]. Accordingly, Chord not only saves a great deal of bandwidth, but also decreases search latency than flooding-based algorithms. However, the Chord cannot offer relatively high performance under mobile environment. In this paper, in order to achieve high hit ratio, low search latency and bearable network traffic for MANET, two kinds of modified Chord algorithms, Backtracking Chord and Redundant Chord, are proposed.

The organization of the paper is as follows. An overview of Chord and its problem in MANET is introduced in Section 2, and proposed algorithms are mentioned in Section 3. In Section 4, simulation and experimental results are discussed, and Section 5 is summarized with conclusion and future work.

2. Overview of Chord algorithm

2.1 Chord

One of the decentralized P2P algorithms, Chord is different from flooding-based search algorithm such as Gnutella[7]. In an N-node Chord system, the wanted data can be retrieved by only sending $O(\log N)$ query messages to $O(\log N)$ nodes. These $O(\log N)$ nodes are called successor nodes. Each data is given a key and the node ID which is derived from hash function. Then each data can be searched by a pair data and key item. For example, in Fig. 1, in order to search data, K19, the query message is sent to successor node, N20, whose identifier is close to K19.

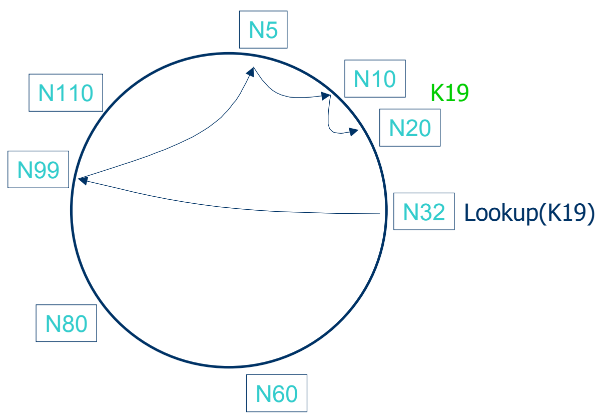

Fig. 1. Chord algorithm

As shown in Table 1, Chord needs less memory space than Napster and less query messages than Gnutella.

Table 1. Memory space and query message of P2P algorithm

P2P algorithm	Memory space	Query message
Napster	$O(N)$	$O(1)$
Gnutella	$O(1)$	$O(N)$
Chord	$O(\log N)$	$O(\log N)$

2.2 The problem of Chord in MANET

We implement simulation with NS2(Network Simulator 2)[8] for existing Gnutella and Chord in MANET. In Fig. 2, the search latency of Chord decreases 74% than Gnutella on average. In terms of the search latency, the result indicates that Chord is more adaptable for MANET than Gnutella[9,10]. However, as shown in Fig. 3, the hit ratio of Chord with mobility is 32% while it is 84% with no mobility on average.

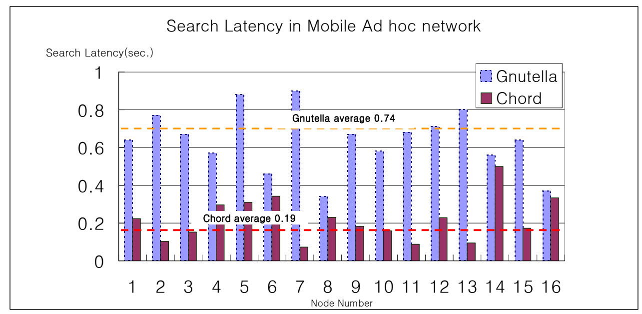

Fig. 2. Search latency of Gnutella and Chord in MANET

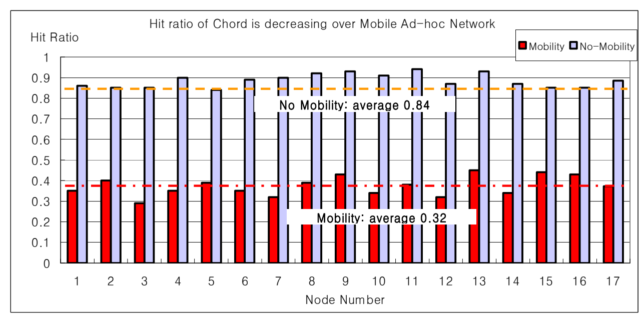

Fig. 3. Hit ratio of Chord in mobility and no mobility environment

The reason for the low hit ratio of Chord in MANET is shown in Fig. 4. The mobile node in MANET, sometimes, leaves the radio region because of its mobility. When the node, N32, sends query to the disappeared successor node, N99, the node, N32, can not continue to search the data.

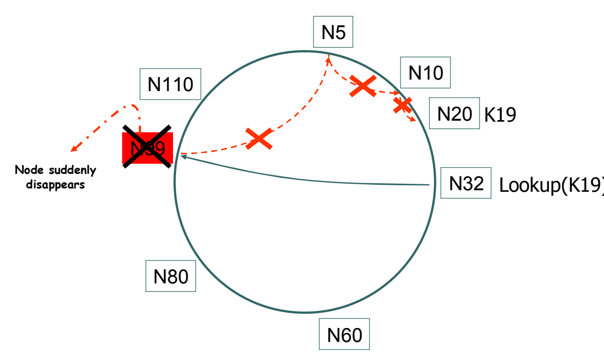

Fig. 4. Chord algorithm in MANET

Therefore, the Chord could not be employed in MANET environment as it is. In this paper, Backtracking Chord and Redundant Chord algorithms are proposed to maintain successor table in case successor node disappears abruptly,.

3. Proposed Algorithms

3.1 Backtracking Chord algorithm

Since the search fails by the disappearance of the requested successor node, in proposed *Backtracking Chord* algorithm, we set time-out to every query search. If time-out occurs, the query is sent to a new successor node instead of stopping the search. The new designated successor node is the node just in front of the disappeared node in the successor table. The table in the right bottom of the Fig. 6 is successor table which contains the successor node lists in it. The number of time-out is set to t and the value of t is between 0 and $\log N$. According to the value t, the number of retransmissions is determined. When a time-out occurs, the query is sent to the next stage successor node. Fig. 5 is the flowchart of Backtracking Chord. If a node joins the mobile network under Backtracking Chord algorithm, node ID is assigned and a successor table is made. Then the node can request data as shown in the flowchart.

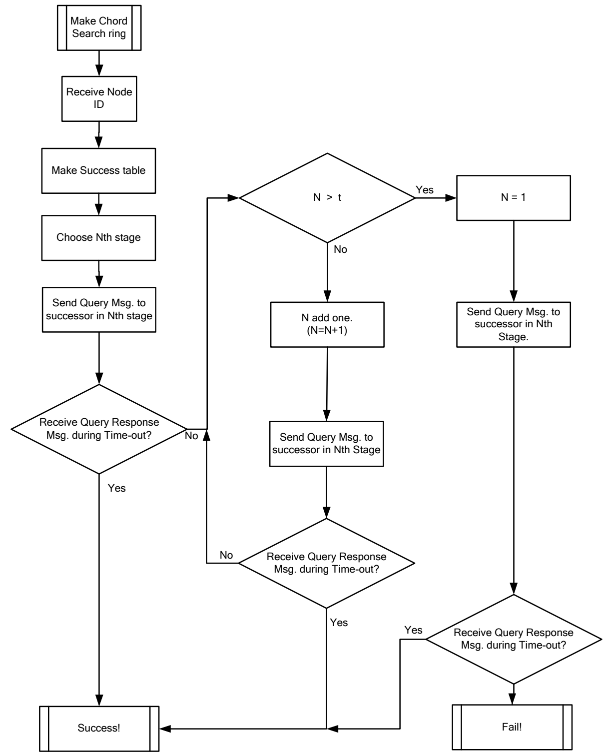

Fig. 5. Flowchart of Backtracking Chord

In Fig. 6, t is set to four, if the search fails(time-out) by the disappearance of the first stage successor node, N99, the query is sent to the second stage successor node, N80, which is just in front of the node, N99, in the successor table. However, if the time-out occurs until t equal to four, then the query is resent to the first stage successor node, N99, again. If the search fails in this time too, then the Backtracking Chord algorithm stops the search.

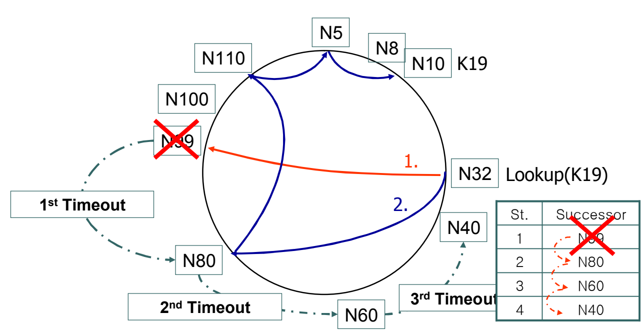

Fig. 6. Backtracking Chord algorithm in MANET

The Backtracking Chord algorithm can increase hit ratio t times in maximum by retransmitting the query according to the value *t*. However, the search latency increases *t* multiples of *time-out* than primitive Chord in maximum. Since no more than $O(t \log N)$ query messages are produced for search, the Backtracking Chord can save bandwidth significantly.

3.2 Redundant Chord algorithm

In *Redundant Chord* algorithm, the requester node sends *r* number of queries to *r* number of successor nodes simultaneously, while Backtracking Chord algorithm sends queries in order. The value *r* is the number of queries sent at once, and the range of *r* is between 0 and $\log N$. In Redundant Chord algorithm $O(r \log N)$ query messages are generated and the search latency is reduced considerably by sending queries concurrently. The flowchart of Redundant Chord is shown in Fig. 7.

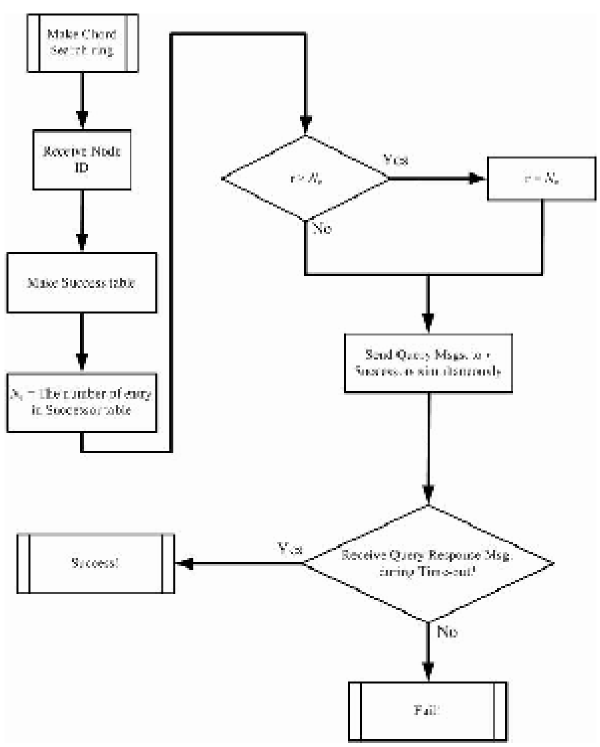

Fig. 7. Flowchart of Redundant Chord

The Redundant algorithm adopts the fastest reply from one of the successor nodes by relaying search via redundant tables of successor nodes. In Fig. 8, when the node, N32, sends queries to four successor nodes, although node, N110, disappears, the node, N32, can get information through N99. Therefore, the probability of obtaining the data is *r* times higher than the existing Chord algorithm with maintaining the search latency similar to Chord. If all the requested successor nodes disappear, the search can not be continued any more and the requester receives timeout message. However, with the increment of the mobile nodes *N*, it is nearly impossible that all the successor nodes disappear concurrently. In a large scale network rather than a small scale network, the algorithm can guarantee a fairly acceptable hit ratio within an appropriate search time. The hit ratio of the Redundant Chord has similar effect to flooding based Gnutella when *r* increases. However, in Redundant Chord, the maximum *r* is $\log N$. Accordingly the Redundant Chord achieves flooding-based like hit ratio via only $O(r \log N)$ query messages.

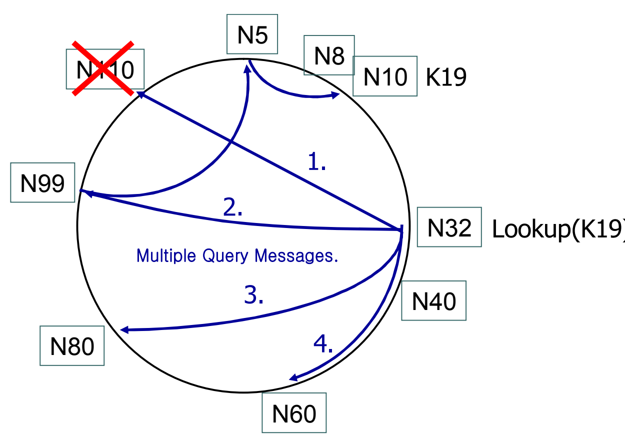

Fig. 8. Redundant Chord algorithm in MANET

4. Simulation and experimental results

4.1 Simulation environment

In this simulation, performance results are acquired by NS2 simulator and one thousand mobile devices are employed(Fig. 9). These mobile nodes move in a radio range according to the random waypoint mobility model in

Glomosim[11]. We select AODV(Ad-hoc On Demand Distance Vector)[12] as Ad-hoc routing protocol, in the simulation. The time-out of Backtracking Chord is given 0.01 second and the Redundant Chord is given 1 second. The radio range of a mobile node is 1m and the area of the simulation environment is 1000 meter wide and 1000 meter length. The number of mobile nodes is one thousand and the mobile nodes move at a speed of 1m/sec. The query messages are generated by random number generator which generates a number from 1 to 20 for every node. However, twenty mobile nodes among one thousand nodes are applied the numbers from 15 to 40 separately. At first, all nodes are in the radio range and then they are set to *on* and *off* state according to the movement property of the selected mobility model. If the node is in the radio range, it is set to *on* state, otherwise it is set to *off* state.

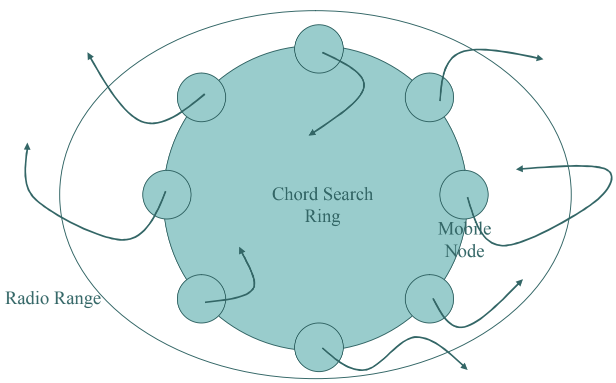

Fig. 9. Simulation model

We compare the hit ratio and the search latency of the proposed algorithms with Gnutella and existing Chord. The hit ratio H is given by

$$H = \frac{M_r}{M_q} \tag{1}$$

Where M_r is the number of response messages and M_q is the number of query messages in individual mobile node. The search latency T_s is given by

$$T_s = T_r - T_q \tag{2}$$

where T_r is the query message timestamp and T_q is the query message timestamp.

4.2 Simulation results

In Fig. 10, hit ratio of Chord and Backtracking Chord in MANET is compared. There are seventeen nodes which request data more frequently than the others are selected among one thousand mobile nodes. The hit ratio of Backtracking Chord presents 72% on average, while hit ratio of existing Chord presents 32%. In this case, *t* is set to two. The hit ratio of node 8 in Backtracking Chord is especially low. Because in node 8, the probability that the two requested successor nodes disappear is higher than the other nodes. The problem can be solved with increasing the value *t* as in Fig. 13.

In Fig. 10 the hit ratio of the Backtracking Chord distributes in wide range, because queries are generated randomly. Therefore some successor nodes suffer from processing overhead and can not handle the query immediately. In this paper, we assumed the processing overhead may be ignorable. It will be solved in next paper. Therefore, Backtracking Chord is quite an efficient search algorithm.

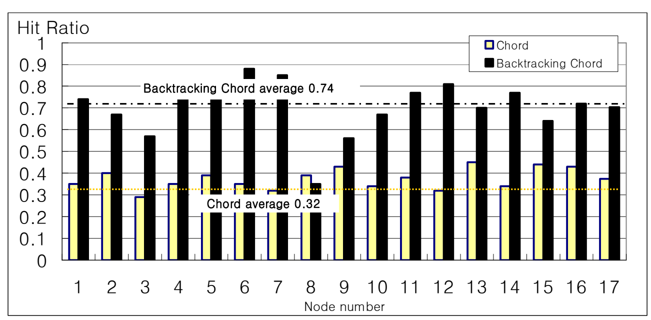

Fig. 10. Hit ratio of Chord and Backtracking Chord in MANET

In Fig. 11, the search latency of Gnutella and Redundant Chord is compared. Sixteen nodes which request the queries more frequently than the others are selected. As in Fig. 11, the search latency of the Gnutella is 0.74 second and the Redundant Chord is 0.39 second on average. The search latency in Redundant Chord is reduced by half of the Gnutella. Since query messages are sent to several successor nodes concurrently in Redundant Chord, the search latency is lower than Gnutella which floods the queries to all of the nodes gradually. The node 10 of Redundant Chord presents high search latency because in this case all of the successor nodes disappears coincidentally, and one of them appears before time-out(1sec) occurs.

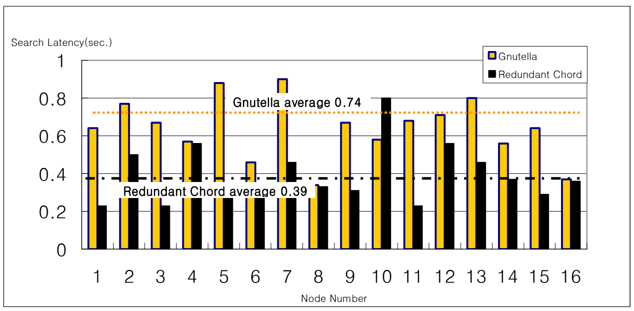

Fig. 11. Search latency of Gnutella and Redundant Chord in MANET

In Fig. 12, the hit ratio of Chord and Redundant Chord is compared in MANET. As shown in Fig. 12, the hit ratio of the Redundant Chord is 53% and it is higher than Chord. Since the value r is two, the probability that these two requested successor nodes disappear is relatively high. As shown in Fig. 14, if the value r increases, the hit ratio improves accordingly.

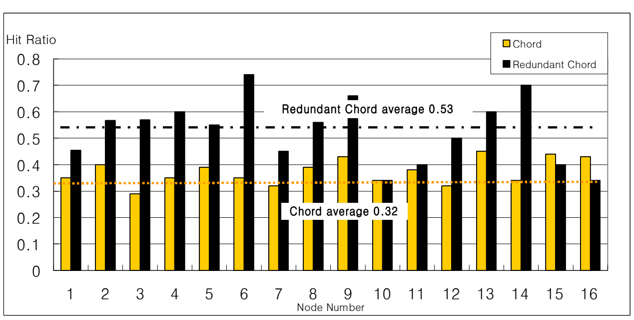

Fig. 12. Hit ratio of Chord and Redundant Chord in MANET

Fig. 13 and Fig. 14 are the correlation among the quantity of query messages, search latency and hit ratio in Backtracking Chord and Redundant Chord respectively.

As we know, with the increment of the value t in Backtracking Chord and of the value r in Redundant Chord, the quantity of messages increases linearly. The simulation result shows that, the hit ratio of both Backtracking Chord and Redundant Chord is increasing according to increment of the value t and r respectively. However, the search latency is different. In Backtracking Chord, it presents linear increment. On the other hand, it decreases in Redundant Chord according to the increment of value t and r separately. The reason is as follows. In Backtracking Chord, the query messages are sent to each successor nodes one by one. For each query, it spends $O(\log N)$ search time. Therefore, in Backtracking Chord, the search time increases linearly. However, in Redundant Chord, the query messages are sent to r number of the successor nodes simultaneously. It prevents the increment of the search latency. Moreover, the routing path of the query message in each successor node is a little different in each successor node and Redundant Chord adopts the fastest reply from them. Consequently, with the increment of the value r, the search latency of the Redundant Chord algorithm decreases.

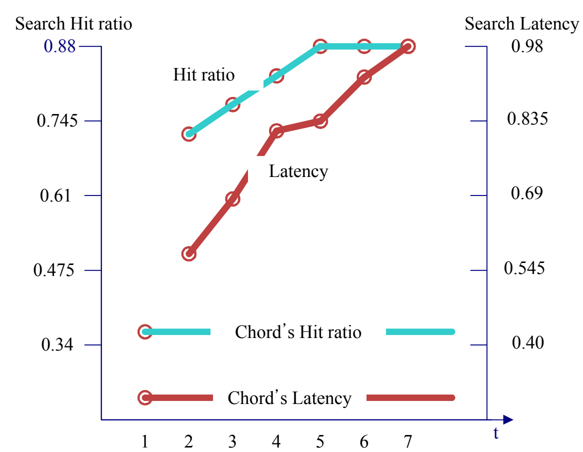

Fig. 13. Convergence of Backtracking Chord

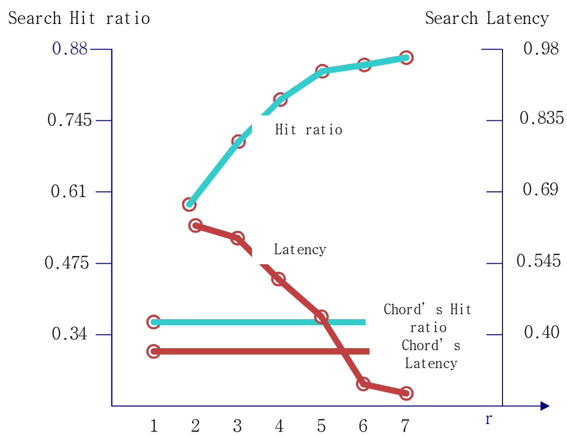

Fig. 14. Convergence of Redundant Chord

In Fig. 15, according to the increase of the value *t*, search latency increases linearly, while hit ratio converges on 88% in Backtracking Chord. The reason of convergence is that each mobile node has different size of radio region in MANET. The overlap of the radio region causes data collision that could not transfer the query messages occasionally. Therefore, the hit ratio could not exceed a certain degree, 88%. In Fig. 15, when the value *t* equals to two, it represents wide range of hit ratio distribution. When the value *t* equals to seven, it shows high hit ratio distribution. If the value *t* increases, the search latency increases accordingly. Therefore, in a small size MANET, the Backtracking Chord can configure a reliable P2P search network.

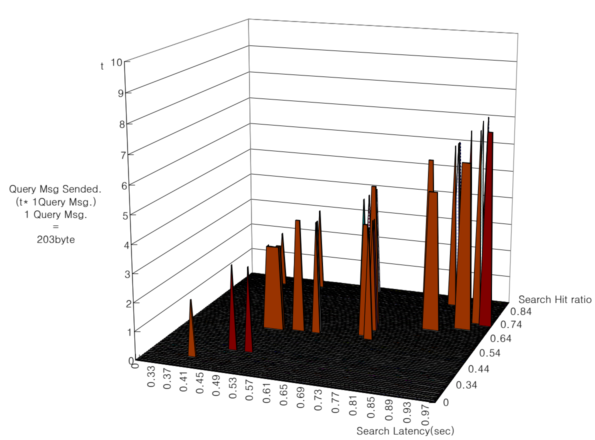

Fig. 15. Correlation of variables in Backtracking Chord

In Fig. 16, according to the increase of the value *r*, the search latency decreases. The reason is that, multiple query messages are sent to multiple nodes and the search is implemented through multiple paths concurrently. It increases the possibility of fast search. However if all the successor nodes disappear, it produces high search latency, and wide range distribution of the hit ratio. In a large scale MANET, a certain hit ratio can be guaranteed and the search latency decreases according to the increase of the value *r*. However, in a small scale MANET, the range of value *r* is limited thus the algorithm may not provide reliable P2P search. Therefore, the Redundant Chord is appropriate for a large size MANET.

Since the hit ratio of existing Chord algorithm in MANET is 37.4% on average, the Chord can not serve as it is. The Backtracking Chord and Redundant Chord algorithms are proposed in this paper, and simulation results prove that these two algorithms are reliable and

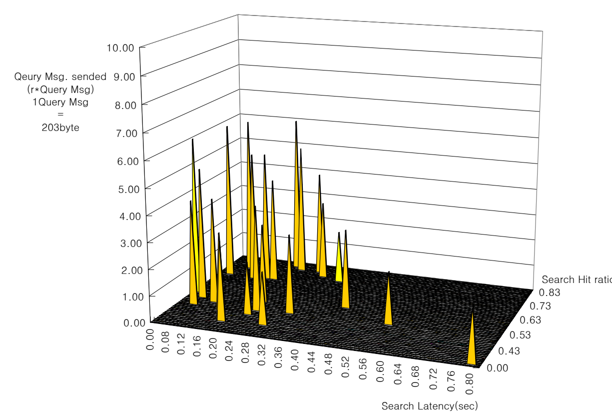

Fig. 16. Correlation of variables in Redundant Chord

efficient in MANET. The hit ratio of the Backtracking Chord is 74% on average and maximum 88%. The hit ratio of the Redundant Chord achieves up to 82% and the search latency of Redundant Chord is reduced by half of the Gnutella in MANET(Table 2).

Table 2. Backtracking Chord vs. Redundant Chord I

	Query msg.	Avg. search latency(sec)	Avg. hit ratio(%)	Network applied
Backtracking Chord	$O(t \log N)$	0.67~1.34	74~88	Small MANET
		proportion with *t*	proportion with *t*	
Redundant Chord	$O(r \log N)$	0.13~0.39	58~84	Large MANET
		inverse-proportion with *r*	proportion with *r*	

5. Conclusion and future work

The P2P file search in mobile environment is in the beginning of the study, it is progressed actively in wired internet. The most considerable problems in mobile network are the constrained bandwidth and mobility. In order to develop a search algorithm appropriate for mobile environment, we verified the existing search algorithms, Gnutella and Chord, in MANET. According to the simulation results, Gnutella generates too much overhead traffic for maintaining network connections. Chord saves bandwidth greatly with $O(\log N)$ query messages for a search compared to Gnutella. However, the hit ratio of the Chord is too low in MANET. Over all, existing P2P query search algorithms consume too much bandwidth and increase search latency to ensure search accuracy in MANET. Therefore, we propose two kinds of algorithms which are modified from the existing Chord algorithm, Backtracking Chord and Redundant Chord. Since the hit ratio of the Backtracking Chord is improved via maintaining appropriate search latency and certain query messages, it resolves bandwidth problems in MANET. The Redundant Chord solves the search latency problem by sending multiple query messages simultaneously, and also provides relative hit ratio in MANET.

In our simulation, Random waypoint mobility model is utilized. However, in this model, the mobile nodes move to a certain direction. In the future work, we will adopt a more flexible mobility model where the nodes move to random direction to realize a real ad-hoc network like environment. The correlation between the number of time-out, *t,* the number of queries sent at once, *r,* and the number of the mobile nodes will be investigated by mathematical expression and simulation.

In this paper, hit ratio includes the error range produced by processing overhead, and also search latency includes the error range caused by processing time. It will be modified via more detailed analytical model in future work.

We are working on a simulation to analyze the bandwidth utilization according to the change of the value *t* in Backtracking Chord, and *r* in Redundant Chord, and also work out the optimal *t* and *r*.

References

[1] Napster, http://www.napster.com/.

[2] Gnutella. http://gnutella.wego.com/.

[3] M. Ripeanu, "Peer-to-Peer Architecture case study: Gnutella Network", Proceedings of the First International Conference on Peer-to-Peer Computing, 2001

[4] Alexander Klemm, Christoph Lindemann, Oliver P. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc networks", Proc. IEEE Semiannual Vehicular Technology Conference, October 2003.

[5] Chord project, http://pdos.lcs.mit.edu/chord/.

[6] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnany:'Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications' MIT Laboratory for Computer Science.(ACM Conf.2002)

[7] Gnutella Protocol draft v0.6.

[8] NS2, http://www.isi.edu/nsnam/ns/

[9]T. H. Hu, B. Thai, A. Seneviratne, "Supporting Mobile Devices in Gnutella File Sharing Network with Mobile Agents", ISCC, Kemer -

Antalya, Turkey, July 2003.

[10] L. Barbosa e Oliveira, I. Guimaraes, A. A. Ferreira Loureiro, "Evaluation of Ad-Hoc Routing Protocols under a Peer-to-Peer Application", Proc. IEEE Wireless Communications and Networking Conference, March 2003.

[11] X. Zeng, R. Bagrodia, M. Gerla, “GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks”, Proceedings of Parallel and Distributed Simulation, 1998

[12] Charles E. Perkins, Elizabeth M. Belding-Royer, and Ian Chakeres. "Ad Hoc On Demand Distance Vector (AODV) Routing." IETF Internet draft, draft-perkins-manet-aodvbis-00.txt, Oct 2003 (Work in Progress),