Mobile P2P Networks for Highly Dynamic Environments

Kei Takeshita
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita
Osaka, 565-0871, Japan
Email: k-takest@ist.osaka-u.ac.jp

Masahiro Sasabe
Graduate School of Engineering
Osaka University
2-1 Yamadaoka, Suita
Osaka, 565-0871, Japan
Email: sasabe@comm.eng.osaka-u.ac.jp

Hirotaka Nakano
Cybermedia Center, Osaka University
1-32 Machikaneyama, Toyonaka
Osaka, 560-0043, Japan
Email: nakano@cmc.osaka-u.ac.jp

***Abstract*—Mobile ad hoc network (MANET) is a wireless network constructed by multiple mobile nodes, which does not rely on any infrastructure. MANET can be used as an emergent network when a disaster occurred. Participants of a meeting, conference, or event can also build a temporal information-sharing network over MANET. Since there is no central server in the network, each node must find out its desired information (objects) by itself. To tackle this problem, some researchers have proposed construction schemes of mobile P2P networks, such as Ekta and MADPastry, by modifying distributed hash table (DHT) schemes. Ekta and MADPastry reduce communication overhead and path length by integrating application-layer routing and network-layer routing. Furthermore, MADPastry introduces a clustering method which groups physically-close nodes to improves system performance. However, it has also been pointed out that the system performance deteriorates in highly dynamic environments. In this paper, we extend MADPastry by adding a method sharing pointers among nodes in the same cluster to solve this problem. Through several simulation experiments, we show that the proposed method improves the success rate up to 40 % compared with MADPastry.**

***Index Terms*—mobile ad hoc network (MANET), distributed hash table (DHT), MADPastry, performance evaluation**

I. Introduction

With the proliferation of mobile nodes, such as laptop PCs, PDAs, and mobile phones, mobile ad hoc network (MANET) have been attracting many users to construct a temporal wireless network in various situations. For instance, MANET can be used as an emergent network for communication among people when a disaster occurred and existing infrastructures failed. In another case, participants of a meeting, conference, or event can also build a temporal information-sharing network over MANET to exchange their own files each other.

In MANET, a source node can communicate with its destination node through a multi-hop path. The path between them is determined by a routing protocol such as DSR [1], AODV [2], or etc. However, these protocols do not provide the source node with the location of its desired information (objects). Broadcast is a simple scheme to find out the object. In broadcast, nodes forward queries to all of their neighbors. Since all nodes in the network are the target of search, the search tends to success with a high probability. However, as the network size becomes large or the number of queries increases, the success rate of search decreases due to packet collisions. To efficiently discover objects with low overheads, some researchers have been proposed construction schemes of mobile P2P networks based on distributed hash table (DHT) which is based on a unicast communication [3], [4]. Typical DHT schemes are Pastry [5], Chord [6], Tapestry [7], and CAN [8]. The main contribution of them is keeping low search cost with the increase of network size. For example, Pasty, Chord, and Tapestry guarantee $O(\log N)$ search costs, that is hop count, for any object. Here, N is the network size.

However, the topological structure of MANET dynamically varies due to node mobility, participation, or departure. It has been pointed out that the success rate of search doesn't improve when DHT is simply constructed on the top of MANET and a cross layer approach is significant [9]. To solve this problem, Ekta and MADPastry integrate Pastry with DSR and AODV to share routing information between network-layer and application-layer. This integration contributes to adaptability to the topological changes in MANET and reduction of communication overheads in the system. Furthermore, MADPastry proposes a clustering method which groups overlay nodes taking into account the corresponding physical distance, that is hop count between nodes in the underlay network. In MADPastry, queries are forwarded in a unicast manner between clusters. On the other hand, nodes can also broadcast queries inside a cluster because the corresponding pointers exist in the same cluster.

It has been pointed out that the successful probability of search deteriorates and overheads to maintain the P2P network increase as node velocities become high [4]. In this paper, we extend MADPastry by adding a method sharing pointers among nodes in the same cluster to solve this problem. Through several simulation experiments, we evaluate how the proposed method is effective compared with MADPastry.

The rest of the paper is organized as follows. In section II, we describe overviews of Pastry and MADPastry. Section III describes problems of MADPastry in highly dynamic environments. Then, we introduce the proposed method and show the effectiveness through simulation experiments in section IV. Finally, we conclude this paper and describe future work in section V.

II. Related Work

In this section, we describe overviews of Pastry and MADPastry.

A. Pastry

Pastry is one of the structured P2P protocols based on DHT. Each node is randomly assigned a unique 128-bit identifier (nodeId) that is generated by a hash function with its IP address and allocated into a circular overlay ID space which ranges from 0 to $2^{128} - 1$. Each object is also assigned a unique 128-bit object identification (key) by using the same hash function to its name and allocated into the same overlay ID space. NodeIds and keys are regarded as a sequence of digits with base 2^b where b is a configuration parameter with typical value 4.

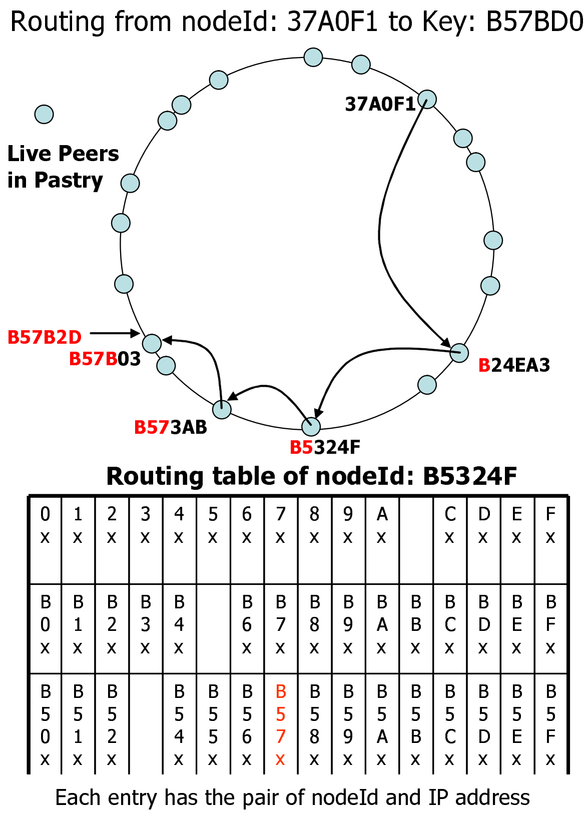
Each Pastry node maintains a routing table, a neighborhood set, and a leaf set. The Pastry's routing table consists of $\log_{2^b} N$ (N is the number of nodes in the network) rows each of which has $2^b - 1$ entries (a pair of nodeId and its IP address), as shown in the lower part of Fig. 1. nth row has entries whose nodeIds share the first $n-1$ digits with the present node's nodeId. Since nth digit has 2^b possible numbers, nth row has $2^b - 1$ entries. The leaf set L is a set of nodes with the $\frac{L}{2}$ numerically closest larger nodeIds, and the $\frac{L}{2}$ nodes with numerically closest smaller nodeIds, relative to the present node's nodeId. The neighborhood set M contains the nodeIds and IP addresses of the M nodes that are closest (according the proximity metric) to the local node. The neighborhood set is not normally used in routing messages; it is used in maintaining locality properties, joining method.

Figure 1 illustrates a routing example. To search a key, each node forwards the query to the node which has the nodeId sharing one more digit with the key based on the routing table. If there is no appropriate entry, the node forwards the query to the node in the neighbor or leaf set which has the nodeId sharing at least one more bit with the key. Therefore, Pastry guarantees the query forwarding unless $\frac{L}{2}$ former or latter nodes in the leaf set fail simultaneously. Since the routing is normally processed every digit, overlay hop count can be expressed as $O(\log_{2^b} N)$. Each node needs to maintain the routing table, neighbor set, and leaf set by pinging to each entry in them.

When the query arrives at a node whose nodeId is the numerically closest to key, the search will success if the node has a pointer about the key (a pair of key and IP address of the node that maintains the corresponding object). We should note here that a registration of a pointer can be accomplished by the object owner using the same mechanism as the query search.

B. MADPastry

MADPastry is an integrated scheme of Pastry and AODV. It introduces the following two kinds of methods.


Routing from nodeId: 37A0F1 to Key: B57BD0
Live Peers in Pastry
37A0F1
B57B2D
B57B03
B24EA3
B573AB
B5324F
Routing table of nodeId: B5324F
Each entry has the pair of nodeId and IP address

0 x	1 x	2 x	3 x	4 x	5 x	6 x	7 x	8 x	9 x	A x		C x	D x	E x	F x
B 0 x	B 1 x	B 2 x	B 3 x	B 4 x		B 6 x	B 7 x	B 8 x	B 9 x	B A x	B B x	B C x	B D x	B E x	B F x
B 5 0 x	B 5 1 x	B 5 2 x		B 5 4 x	B 5 5 x	B 5 6 x	B 5 7 x	B 5 8 x	B 5 9 x	B 5 A x	B 5 B x	B 5 C x	B 5 D x	B 5 E x	B 5 F x

Fig. 1. Pastry routing table

1) Updating information by packets overheard at nodes: In MADPastry, each packet contain AODV sequence number, nodeId, and IP address for each last node in the Pastry and AODV routings. Whenever a node overhears or receives any packet, it updates its AODV routing table, Pastry routing table, and leaf set. It contributes to reducing maintenance overheads and achieving high adaptability to the environmental change.

2) Clustering nodes taking into account their physical locations: MADPastry associates the node's physical location with its overlay's location. Specifically, a cluster is formed with physically-close nodes by coordinating the first digit of their nodeIds with that of the cluster head. The determination of the cluster head is described in the next paragraph. Since the Pastry routing is processed every digit, physical hop count can be reduced compared with the original Pastry as shown in Fig. 2. Note that the overlay hop count does not change.

Since there is no central server in MANET, cluster heads must be elected in a fully-distributed manner. MADPastry uses $landmark\ key$ to form a cluster. It generates K $landmark\ keys$ that evenly divides the overlay ID space into K sub spaces. For instance, in case of $K = 16\ (= 2^b)$, $landmark\ keys$ become 0800...000, 1800...000, . . . , E800...000, F800...000. A node whose nodeId is the numerically closest to $landmark\ key$ becomes a cluster head and starts to periodically broadcasting a cluster-head beacon to all nodes in its cluster.

Whenever a node overhears or receives a cluster-head bea-

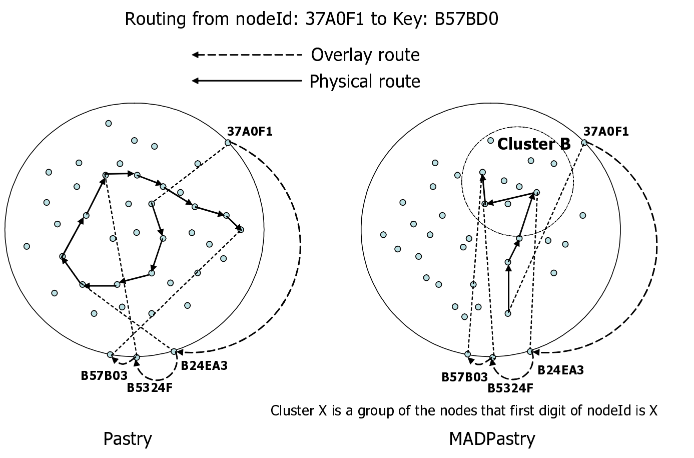

Routing from nodeId: 37A0F1 to Key: B57BD0
Overlay route
Physical route
37A0F1
B57B03
B5324F
B24EA3
Cluster B
Cluster X is a group of the nodes that first digit of nodeId is X
Pastry
MADPastry

Fig. 2. MADPastry clustering

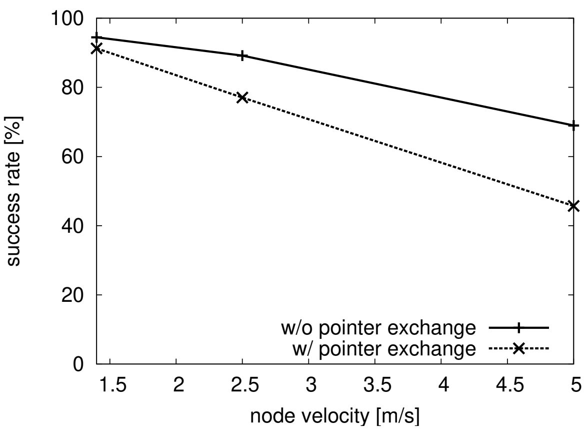


Fig. 3. Success rate of MADPastry

con, it stores the current cluster head's nodeId and the physical hop count to the cluster head. Nodes periodically check the closest cluster and move to it if it is different from the current cluster. Since it has to change the first digit of its own nodeId, changing cluster requires leaving and re-joining MADPastry network.

Although the routing among different clusters is the same as that in Pastry, MADPastry uses the leaf set for the routing inside a cluster. If the leaf set includes a node having closer nodeId to key and an AODV route to be reached, the corresponding query is forwarded to the node. Otherwise, the query is broadcasted inside the cluster. These mechanism accomplishes effective searching taking into account the physical proximity among nodes inside a cluster.

III. Problems of MADPastry in highly dynamic environments

It has been pointed out that the successful probability of search deteriorates as node velocities become high [4]. The main reason is that queries tend to be lost due to the disappearances of AODV routes. When the node velocities increase, it is difficult for nodes to maintain the AODV routing table consistent with the real topology. According to IEEE802.11 standard, a node continues sending a packet until it receives an acknowledgement or the number of transmissions reaches the pre-determined threshold. If the node failed to send the packet to the next-hop node in the AODV route, it checks the position in the current AODV route. If it is located on the former part of the route, it abandons sending the packet that means the query is lost. Otherwise, it tries to find another AODV route to the destination with the overhead of broadcasting.

In addition, they did not accurately consider the overheads and risks of cluster changes. Since each node is responsible for pointers whose $keys$ are the closest to its nodeId, it must update pointers in accordance with response to a cluster change. However, such pointer updating may also fail due to the disappearances of AODV routes.

Periodic re-registration of pointers by their corresponding object holders seems to reduce the pointer disappearances from the network. However, the pointer registration is conducted by the same mechanism as query routing and may also fail.

We conducted simulation experiments to evaluate how the above mentioned problems affected the search fail. We modified the source code of MADPastry that was written as a module of ns-2 [10] and was provided by the authors [4]. Nodes moved in accordance with random waypoint model [11] with *pause time* of 0 [sec] and velocity of 1.4, 2.5, and 5.0 [m/s], respectively. At the start of simulations, 250 nodes were randomly allocated on a two-dimensional square space whose node density was 100 [nodes/Km^2]. MAC layer was IEEE802.11 with transmission rate of 11 [Mbps] and transmission range of 250 [m]. 1000 pointers were uniformly allocated into Pastry network. Thus, each node maintained four pointers in average. Each node sent a query to an object randomly chosen from them at interval of 10 [sec]. In the case that nodes conducted pointer exchanging (*w/ pointer exchange*), object holders re-registered pointers relevant to their objects at interval of 120 [sec] to reduce the disappearances of pointers from the network. Note that the pointer exchange was achieved by adding pointers to leave and join messages.

Like Ref. [4], the success rate of search is defined as the ratio of the number of queries reaching nodes whose nodeIds are the closest to $keys$ to the whole number of queries if pointer exchanging is ignored (*w/o pointer exchange*). On the other hand, we define the success rate as the ratio of the number of queries reaching nodes that have the corresponding pointers to the whole number of queries in the case of *w/ pointer exchange*. The simulation time was 3600 [sec] and we show the average of the latter 2000 [sec] simulation in the following results.

Figure 3 illustrates the success rate of both cases: *w/o pointer exchange* and *w/ pointer exchange*. Note that the results of *w/o pointer exchange* are the same as shown in Ref. [4]. In the case of *w/o pointer exchange*, the success rate shows a depreciation of 30 % at the maximum. This is mainly caused by query disappearances above mentioned. On the contrary, we find that additional 3-25 % deterioration occurs, in the case of *w/ pointer exchange*. This is because some pointers vanish in the processes of cluster changes. We also find that the periodic re-registration of pointers cannot sufficiently suppress the deterioration of success rate. These results indicate that it is difficult for nodes to reliably exchange information with others in highly dynamic environments. In the following section, we introduce a simple but effective solution for this problem.

IV. Resilient Method for Highly Dynamic Environments

In highly dynamic environments where nodes move around quickly, it is difficult to avoid disappearances of AODV routes that make queries and pointers lost. In other words, reliable hop-by-hop data transfer cannot be achieved in such situations. In this section, we alternatively propose a method sharing pointers among nodes in the same cluster. This is an application-level method to cope with query disappearances inside clusters and pointer disappearances. Through simulation experiments, we show the effectiveness of the proposed method. We should note here that query disappearances among clusters cannot be solved by this method. As future work, we plan to modify AODV routing protocol to tackle this problem.

A. Sharing Pointers among Nodes in the Same Cluster

Each node sends its own pointers to other nodes in the same cluster by slightly modifying the periodic beacon messages in MADPastry. Once receiving or overhearing the beacon message from other cluster members, the node stores the pointers. Consequently, we can achieve sharing pointers among nodes in the same cluster. This simple method is effective in terms of both search efficiency and traffic overheads as follows.

- Search efficiency
  Since nodes in the same cluster are physically close, they have chances to receive or overhear queries reaching other destinations. If they possess the pointer relevant to the overheard or received query, they reply to the query instead of the destination. This not only increases the success rate but also shorten the search response. Furthermore, multiplying pointers can reduce the pointer disappearances.
- Traffic overheads
  MADPastry originally has a beacon mechanism in which each node periodically sends a beacon to other nodes in the same cluster. This is essential to autonomously form clusters in the network. The proposed method can be accomplished by only adding pointer information to this beacon message. The beacon size increases with the growth of the number of objects. However, we expect that the number of objects is not so large in realistic situations.

B. Simulation Experiments

We conducted simulation experiments to evaluate the effectiveness of the proposed method. We used the same configurations described in section III. We also focused on the performance of broadcast because it has been pointed out that broadcast was more effective than MADPastry in the highly dynamic environments. In broadcast, nodes forward queries to all of their neighbors except for cases that they have the corresponding objects or received the queries before.

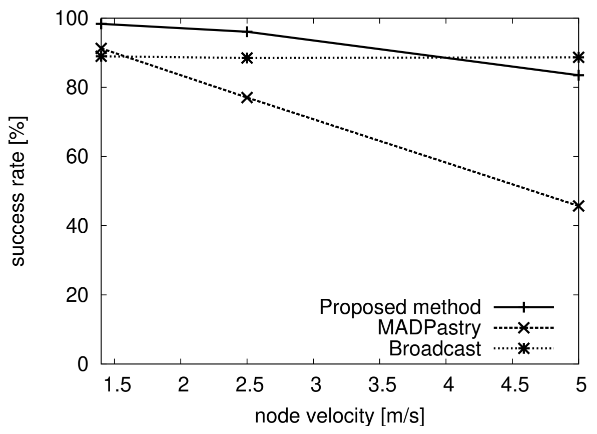


Fig. 4. Success rate

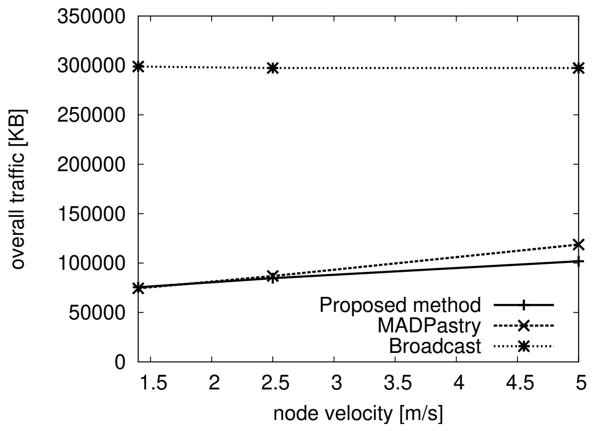


Fig. 5. Oveall traffic

Figure 4 depicts the success rate of the proposed method, MADPastry with pointer exchange, and broadcast. We find that the effect of pointer sharing increases as the node velocity becomes high. Specifically, the proposed method can achieve up to 40% improvement compared with MADPastry. Furthermore, the overall traffic of the proposed method becomes lower than that of MADPastry as the node velocity increases (Fig. 5). Although the proposed method increases the beacon size, the pointer sharing contributes to reduction of query forwarding inside the cluster.

On the contrary, the success rate of the proposed method is about 6 % lower than that of broadcast when the node velocity is 5 [m/s]. However, broadcast requires at most third times as much overall traffic as the proposed method to achieve almost the same success rate.

V. Conclusion and Future Work

MADPastry is one of DHT substrates for MANETs and can achieve high success rate with small overall traffic in relatively stable environments. In this paper, we extended MADPastry by adding a method sharing pointers among nodes in the same cluster to adapt high node mobility. Through simulation experiments, we showed that the proposed method could improve the success rate up to 40 % compared with MADPastry while suppressing the volume of the overall traffic.

As future work, we would like to focus on load balancing among nodes. In MADPastry, *landmark keys* are uniformly distributed in the overlay network. However, each cluster size depends on the physical node distribution. If we can equalize the cluster size, the fairness in terms of the number of processing queries per node can be improved and the broadcast traffic can be further suppressed.

References

[1] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in *Mobile Computing*, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.

[2] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. Washington, DC, USA: IEEE Computer Society, Feb. 1999, pp. 90–100.

[3] H. Pucha, S. M. Das, and Y. C. Hu, “Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks,” in *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, Dec. 2004, pp. 163–173.

[4] T. Zahn and J. Schiller, “Madpastry: A dht substrate for practicably sized manets,” in *Proceegings of the Fifth Workshop on Applications and Services in Wireless Networks*, Paris, France, June 2005.

[5] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. London, UK: Springer-Verlag, Nov. 2001, pp. 329–350.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, Aug. 2001, pp. 149–160.

[7] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An infrastructure for fault-tolerant wide-area location and routing,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-01-1141, Apr. 2001.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, “A scalable content-addressable network,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, Aug. 2001, pp. 161–172.

[9] J. Wu, *Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks*. Auerbach Publications, 2006, ch. 47.Peer-to-Peer Overlay Abstractions in MANETs, pp. 845–862.

[10] “The network simulator ns-2,” http://www.isi.edu/nsnam/ns/.

[11] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.