PAPER Special Section on Peer to Peer Networking Technology

Improving Success Ratio of Object Search in Highly-Dynamic Mobile P2P Networks

Kei TAKESHITA[†a)], ***Nonmember***, **Masahiro SASABE**[††b)], ***and*** **Hirotaka NAKANO**[†††c)], ***Members***

SUMMARY Mobile Ad Hoc Networks (MANETs) are temporal and infrastructure-independent wireless networks that consist of mobile nodes. For instance, a MANET can be used as an emergent network for communication among people when a disaster occurred. Since there is no central server in the network, each node has to find out its desired information (objects) by itself. Constructing a mobile Peer-to-Peer (P2P) network over the MANET can support the object search. Some researchers proposed construction schemes of mobile P2P networks, such as Ekta and MADPastry. They integrated DHT-based application-layer routing and network-layer routing to increase search efficiency. Furthermore, MADPastry proposed a clustering method which groups the overlay nodes according to their physical distance. However, it has also been pointed out that the search efficiency deteriorates in highly dynamic environments where nodes quickly move around. In this paper, we focus on route disappearances in the network layer which cause the deterioration of search efficiency. We describe the detail of this problem and evaluate quantitatively it through simulation experiments. We extend MADPastry by introducing a method sharing objects among nodes in a cluster. Through simulation experiments, we show that the proposed method can achieve up to 2.5 times larger success rate of object search than MADPastry.

key words: *mobile ad hoc network (MANET), distributed hash table (DHT), clustering, local information sharing*

1. Introduction

With the proliferation of mobile nodes, such as laptop PCs, PDAs, and mobile phones, mobile ad hoc networks (MANETs) have been attracting many users to construct temporal wireless networks in various situations. For instance, a 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 a MANET to exchange their own information each other.

In a 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], and OLSR [3]. However, these protocols do not provide the source node with the location of its desired information (objects). Broadcast used in Gnutella [4] is a simple scheme to find out the object. In broadcast, nodes forward queries to all of their neighbors, which means that broadcast does not rely on any routing protocol. Since all nodes in the network are the targets of the search, the search seems to success with a high probability. However, as the network size becomes large or the number of queries increases, the success ratio of object search decreases due to packet collisions.

Conti et al. proposed an optimization of Gnutella protocol for the use in MANETs [5] by integrating it with OLSR. This is one of the cross-layer approaches between application layer and network layer. The cross-layer approach achieves a good performance compared with Gnutella. However, there still remains a problem of high traffic load caused by broadcast with an increase of the network size.

To efficiently discover objects over MANETs with low search overheads, some researchers proposed construction schemes of mobile P2P networks based on distributed hash table (DHT) which can enable a unicast-based object search [6]–[10]. Typical DHT schemes are Pastry [11], Chord [12], Tapestry [13], etc. The main contribution of them is keeping low search costs with an increase of the network size. More precisely speaking, they guarantee $O(\log N)$ search costs, namely hop count, for any object. Here, N is the network size.

However, the topological structure of a MANET dynamically varies due to node mobility, participation, or departure. It has been pointed out that the success ratio of object search doesn't improve when a DHT substrate is simply constructed on the top of a MANET and a cross-layer approach is significant [6]. To solve this problem, Ekta [6], CrossROAD [10], and MADPastry [7] integrate Pastry with DSR, OLSR, and AODV respectively to share routing information between network layer and application layer. On the other hand, Caesar et al. proposed virtual ring routing (VRR) that is a network-layer routing protocol inspired by overlay routing protocol based on DHT. In VRR, a node conducts unicast-based object search by forwarding queries only to physically one-hop neighbors. Thus, VRR can significantly reduce the traffic overhead compared with broadcast-based schemes.

These techniques contribute to adaptability to the topological changes in a MANET and reduction of communication overheads in the system. However, Ekta and VRR con-

Manuscript received March 24, 2008.
Manuscript revised July 25, 2008.

[†]The author is with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
[††]The author is with the Graduate School of Engineering, Osaka University, Suita-shi, 565-0871 Japan.
[†††]The author is with the Cybermedia Center, Osaka University, Toyonaka-shi, 560-0043 Japan.
a) E-mail: k-takest@ist.osaka-u.ac.jp
b) E-mail: sasabe@comm.eng.osaka-u.ac.jp
c) E-mail: nakano@cmc.osaka-u.ac.jp
DOI: 10.1093/ietcom/e91–b.12.3851

struct a DHT substrate without taking into account physical distance between nodes, namely hop count between them in the underlying physical network. This causes undesirable long search latency and deterioration of success ratio of object search. In this paper, we demonstrate that the following two methods can enhance the success ratio of object search while suppressing the traffic overheads: a construction method of an overlay network with consideration of the underlying physical network and a sharing method of objects among physically-close overlay nodes. The key idea of both methods is the same, namely reducing physical hop count to search for an object.

As the first method, we adopt MADPastry that proposes a clustering method which groups overlay nodes by taking into account the corresponding physical distance. In MADPastry, queries are forwarded in a unicast manner between nodes belonging different clusters. On the other hand, nodes combine the unicast-based object search with a broadcast-based one limited inside a cluster to quickly find out the desired objects with relatively low search costs because the information on corresponding objects exist in the cluster.

We further propose a method to share objects among nodes in a cluster as the second method. Although the detail of the method will described in Sect. 4, the object sharing method can be easily realized by extending the beacon mechanism used in the clustering method while suppressing system overheads. Through simulation experiments, we demonstrate that MADPastry is not sufficient to achieve high system performance but the combination of MADPastry and the object sharing method drastically improves the system performance.

The rest of the paper is organized as follows. In Sect. 2, we describe overviews of related work. Section 3 reveals the problems of MADPastry in highly dynamic environments. Then, we introduce the proposed method and show the effectiveness through simulation experiments in Sect. 4. Finally, we conclude this paper and describe future work in Sect. 5.

2. Related Work

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

2.1 Pastry

Pastry is one of the structured P2P networks 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. Then, it is 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 (objectId) 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 node maintains any pointer (a pair of objectId and IP address that stores the object) whose objectId is the numerically closest to its nodeId.

Each Pastry node maintains a routing table 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). n-th row has entries whose nodeIds share the first $n - 1$ digits with the present node's nodeId. Since n-th digit has 2^b possible numbers, n-th 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.

To search for 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*. Since the routing is normally processed every digit, overlay hop count can be expressed as $O(\log_{2^b} N)$.

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 objectId 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.

2.2 AODV and Its Extended Versions

AODV is a reactive routing protocol in the network layer. In the reactive routing protocol, a source node starts to establish a route to its destination node when it requires to send data to the destination. Consequently, the reactive routing protocol requires time to discover a route before data transmission but also has adaptability to changes in the physical topology.

An AODV routing table consists of multiple entries each of which indicates a route to a destination node. When a source node sends a packet to a destination node, it searches for the corresponding entry in its routing table. If it already has the entry, it sends the packet to the next-hop node designated in the entry. Otherwise, a *route nonexistence* occurs and the source node tries to discover a new route to its destination node by broadcasting messages to discover (route discovery messages). Any node who has an entry of the destination node replies to the route discovery message. The behavior of a relay node is the almost same as that of the source node. Note that the relay node abandons sending the packet if it has no entry for sending the packet.

Even if a node has an entry for the destination node, it cannot necessarily send the packet to the next-hop node due to a link disconnection that means the next-hop node does not exist in its wireless transmission range. We call this problem as a *route disappearance*. If the route disappearance occurred, the node checks the position in the current AODV route. If it is located on the former part of the

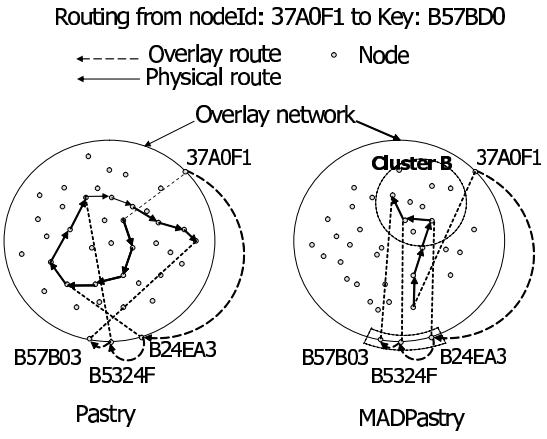

Fig. 1 Routing example: Pastry vs. MADPastry.

route, it abandons sending the packet. Otherwise, it tries to discover a new route to the destination node.

There are several studies on reducing the route disappearances, such as AOMDV [14] and AODV-BR [15]. They prepare alternative routes against the route disappearances by storing all replies instead of only the fastest reply during the route discovery. However, since the second or later discovered routes are not reliable compared to the first discovered route, the effect of storing multiple routes is not so much higher.

2.3 MADPastry

MADPastry is an integrated scheme of Pastry and AODV. It introduces the following two key ideas.

2.3.1 Updating Information by Overhearing Packets at Nodes

In MADPastry, each packet contains AODV sequence number, nodeId, and IP address for each last node in the Pastry and AODV routing. 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 increasing adaptability to the changes in the physical topology.

2.3.2 Clustering Nodes Taking into Account Their Physical Locations

MADPastry associates the node's physical location with its overlay's location. A cluster is formed with physically-close nodes by coordinating the first digit of their nodeIds with that of the cluster head. Since the Pastry routing is processed every digit, physical hop count can be reduced compared with the original Pastry as shown in Fig. 1. Note that the hop count in the overlay routing does not change.

Since there is no central server in a MANET, cluster heads must be elected in a fully-distributed manner. MADPastry uses *landmark keys* to form clusters. It generates K *landmark keys* that evenly divide the overlay ID space into K sub spaces. For instance, for $K = 16$ $(= 2^4)$, *landmark keys* become 0800...000, 1800...000, $\cdots$, E800...000, F800...000.

A node whose nodeId is the numerically closest to a *landmark key* becomes a cluster head and starts periodically broadcasting a cluster-head beacon to all nodes in its cluster, called cluster members. Whenever a node overhears or receives a cluster-head beacon, it records the cluster head's nodeId and the physical hop count to the cluster head that are included in the cluster-head beacon. A node periodically derives the closest cluster head from the recorded list of cluster heads. If the closest cluster head changes, the node moves to the cluster managed by the closest cluster head. Since it has to change the first digit of its own nodeId, changing cluster forces the node into leaving and re-joining MADPastry network.

Although the routing cross over 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 a closer nodeId to *key* and a reachable AODV route, the query is forwarded to the corresponding node. Otherwise, the query is broadcasted inside the cluster. This mechanism accomplishes effective search using the physical proximity among cluster members.

3. Performance Evaluation of MADPastry under Highly Dynamic Environments

3.1 Packet Disappearances Caused by Node Mobilities

It has been pointed out that the success ratio of object search deteriorates as node velocities become high [7]. The main reason is that packets tend to be lost due to the route nonexistence or route disappearances described in Sect. 2.2. Since a lost packet is part of a query message or a message to exchange pointers, two kinds of disappearances occur: query disappearances and pointer disappearances.

- Query disappearances
  Query disappearances are further classified into the following two types.
  - Query disappearances in routing between clusters
    Queries are forwarded in a unicast manner between clusters. Since nodes belonging to different clusters are physically distant each other (Fig. 1), the possibility of query disappearances tends to increase in accordance with relatively large hop count between them.
  - Query disappearances in routing inside a cluster
    Queries are also forwarded in a unicast manner inside a cluster if the corresponding AODV route exists. Otherwise, they are broadcasted inside a cluster with the risk of packet collisions arising from traffic load.
- Pointer disappearances

In Ref. [7], the authors 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 every cluster change. However, such pointer updating may also fail due to the route nonexistence or route disappearances. 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 the query routing and may also fail.

3.2 Simulation Experiments

We conducted simulation experiments to evaluate how the above mentioned problems affected the search failure. We modified the source code of MADPastry that was written as a module of ns-2 [16] and was provided by the authors of Ref. [7]. We used the same parameter settings as Ref. [7]. We set the Pastry parameter b as 4 that means an overlay ID space is denoted as hexadecimal, and the MADPastry parameter K which decides the number of clusters as 16. Nodes moved in accordance with random waypoint model [17] with pause time of 0 sec and speed 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 objects are distributed into the Pastry network such that they evenly divide the overlay ID space. Thus, each node is responsible for maintaining four pointers. Each node sent a query for finding out an object randomly chosen from the entire objects at intervals of 10 sec. In what follows, we call this interval a query interval. In 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 of Pastry.

Like Ref. [7], the success ratio of object 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 not considered (*w/o pointer exchange*). On the other hand, we define the success ratio of object search as the ratio of the number of queries reaching nodes that have the corresponding pointers to the whole number of queries in 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 2 illustrates the success ratios of object search in 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. [7]. In case of *w/o pointer exchange*, the success ratio shows a depreciation of 30% at the maximum. This is mainly caused by query disappearances. On the contrary, we find that additional 3–25% deterioration occurs in case of *w/ pointer exchange*. This is because some pointers vanish during the processes of cluster changes. We also find that the periodic re-registration of pointers cannot sufficiently suppress the deterioration of success ratio of object search. These results indicate that it is difficult for nodes to reliably exchange information with others in highly dynamic environments. In addition, the node originating the query has to retrieve the object based on the pointer information, however, MADPastry did not evaluate this point.

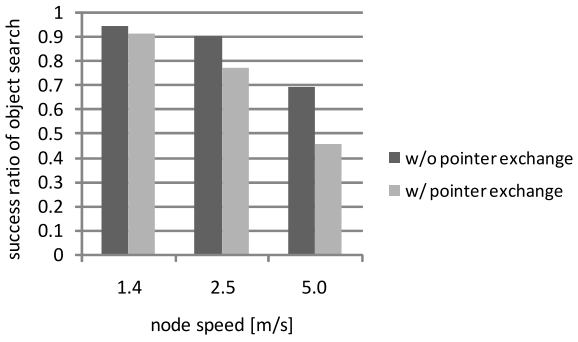

Fig. 2 Success ratio of object search of MADPastry when pointer exchanging is considered.

4. Sharing Objects among Nodes in a Cluster

In highly dynamic environments where nodes move around quickly, it is difficult to completely avoid AODV route nonexistence and route disappearances 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 to share objects among cluster members which is an application-level method to cope with query disappearances inside clusters and object disappearances. The proposed method enables a node to retrieve its desired object through one round trip messaging differently from original MADPastry. Note that the object sharing incurs additional overheads into the system with an increase of the object size. We evaluate the trade-off between the overheads and the system performance in Sect. 4.2.2.

4.1 Detail of the Method

Each node sends its own objects (not pointers) to other cluster members by slightly modifying the periodic beacon message of MADPastry. Once receiving or overhearing the beacon message from other cluster members, the node stores the objects in the message. Then, if the received beacon is originated from the cluster head, the node forwards it to the neighboring nodes to form the cluster. Otherwise, it discards the beacon. Consequently, we can achieve object sharing among cluster members. This simple method is effective in terms of both improvement of search efficiency and reduction of traffic overheads as follows.

- Improvement of search efficiency
  Since cluster members are physically close each other, each cluster member has chances to receive or overhear queries reaching all other cluster members. If it possesses the object relevant to the overheard or received query, it replies to the query instead of the destination node. This not only increases the success ratio of object search but also shorten the response time of the search. Furthermore, multiplying objects can reduce the possibility of object disappearances.
- Reduction of traffic overheads
  MADPastry originally has a beacon mechanism in which each node periodically sends a beacon to other cluster members. This is essential to autonomously form clusters in the network. The proposed method can be accomplished by only adding objects to this beacon message. The beacon size increases with the growth of the number of objects and the object size. However, the number of objects and the object size seem to be relatively small because the mobile P2P network is mainly used as a temporal information sharing network as mentioned in Sect. 1.

Since the object sharing method distributes replicated objects into the network, there is a possibility that duplicate responses to the same query occur. The duplicate responses can be avoided by caching the received or overheard responses to the query at each node.

Replication also causes the consistency problem among the replicated objects. The inconsistency among replicated objects occurs when the owner of the corresponding object updates the object. However, the effect of this inconsistency is small because the object is shared only among the original object holder and its neighboring nodes and the renewal can be automatically detected at the neighbors through the beacon exchange mechanism.

4.2 Simulation Experiments

We conducted simulation experiments to evaluate the effectiveness of the proposed method. We used the same configurations described in Sect. 3 except for the following: Each node did not maintain pointers but objects corresponding to its overlay ID in MADPastry. We assumed text-based information sharing in emergent situations and set the default object size to 250 bytes. We also evaluated the performance of broadcast because it achieves higher success ratio of object search than MADPastry in the highly dynamic environments [7]. In broadcast, nodes forward queries to all of their neighbors except when they have the corresponding objects or have already received or overheard queries to the corresponding objects.

The success ratio of object search is defined as the ratio of the number of queries reaching nodes that have the corresponding objects to the whole number of queries[†]. We omit the phase of sending object back to the nodes originating the corresponding queries because it is the same between the three methods. Overall traffic is defined as the total amount of bytes that the MAC layer receives from the network layer at all nodes. Thus, it includes not only the query traffic but also the maintenance traffic at both application and network layers.

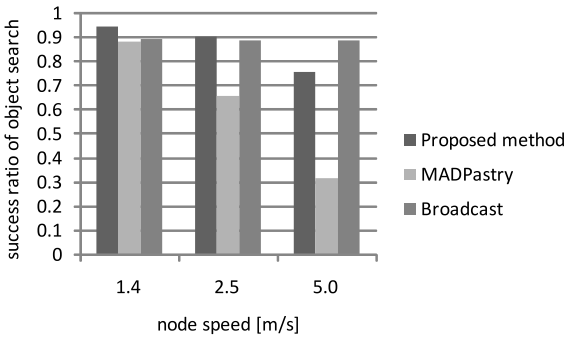

Fig. 3 Success ratio of object search (proposed method, MADPastry, broadcast).

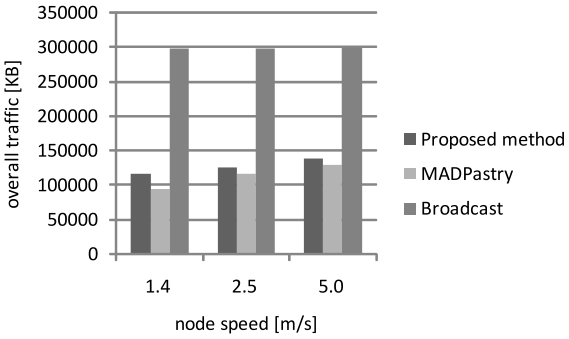

Fig. 4 Overall traffic (proposed method, MADPastry, broadcast).

4.2.1 Basic Performance

Figures 3 and 4 depict the success ratios of object search and the overall traffic in case of the proposed method, MADPastry with object exchange, and broadcast, respectively. We find that the effect of object sharing increases as the node speed becomes high. Specifically, the proposed method can achieve up to 2.5 times larger success rate of object search than MADPastry. The overall traffic of the proposed method slightly increases compared with that of MADPastry because the proposed method shares objects among neighboring nodes.

The success ratio of object search of the proposed method is about 16% lower than that of broadcast when the node speed is 5.0 m/s. However, broadcast requires at least twice as much overall traffic as the proposed method to achieve almost the same success ratio of object search.

[†]This criterion does not take into account the superiority of broadcast to other two methods in terms of partial matching search. However, MADPastry and the proposed method can cope with the partial matching by combining them with the existing methods tackling the partial matching in DHT [18], [19].

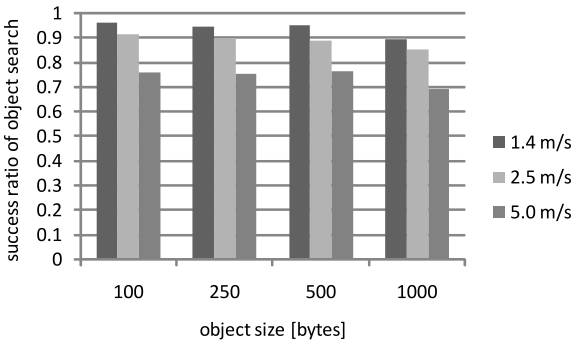

Fig. 5 Success ratio of object search when change object size.

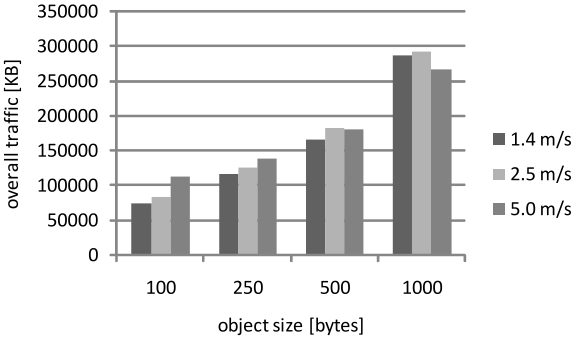

Fig. 6 Overall traffic when changing object size.

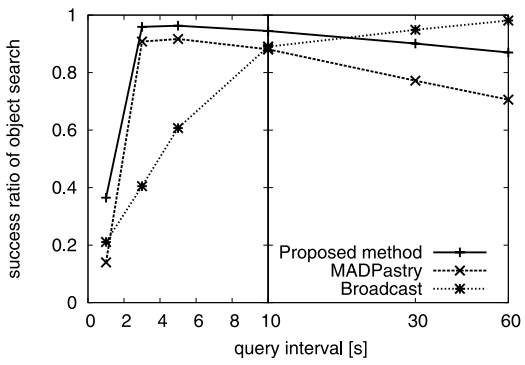

Fig. 7 Success ratio of object search when changing query interval.

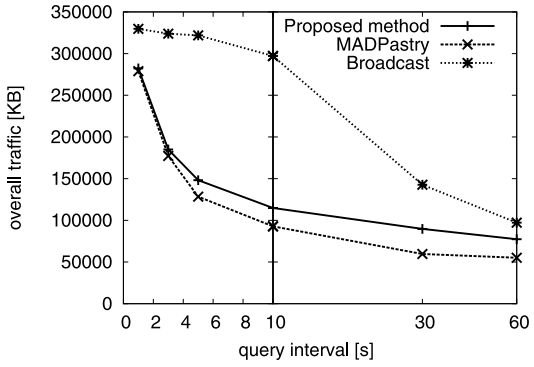

Fig. 8 Overall traffic when changing query interval.

4.2.2 Effect of Object Size

In the proposed method, the overhead of object sharing increases as the object size rises. Figures 5 and 6 illustrate how the success ratio of object search and the overall traffic change when the object size is set to 100, 250, 500, and 1000 bytes, respectively.

If the object size is below 500 bytes, the success ratio of object search does not almost change for each node speed. On the contrary, the overall traffic linearly rises with an increase of the object size and reaches almost the same volume as the broadcast (Fig. 4) at the object size of 1000 bytes. Thus, we can conclude that the proposed method is more effective than broadcast if the object size is less than 1000 bytes. An example of situations suitable for the proposed method is information sharing in an emergency where text-based information exchanges for safety confirmation and condition report occupy the large portion of traffic.

4.2.3 Effect of Request Rate

To reveal the performance of the MADPastry, proposed method, and broadcast when changing the load on the system, we ran simulations by setting the node speed to 1.4 m/s and the query interval to 1, 3, 5, 10, 30, and 60 sec. The other parameters were the same as those in Sect. 3.

Figures 7 and 8 depict the success ratio of object search and the overall traffic of the three methods when the query interval is changed. In case of the proposed method and MADPastry, the success ratio of object search becomes the maximum between three and ten seconds and gently deteriorates in the range of more than ten seconds. Since an increase of the overall traffic gives nodes more chances to update their AODV and Pastry routing tables, the risk of link disconnections is reduced. However, the success ratio of object search drastically deteriorates when the query interval is set to 1 sec. This is because the system is under a heavy congested state as shown in Fig. 8. Such a heavy congested state is also found when the query interval is below 10 in case of broadcast. Broadcast is effective when the query interval is over 10 while the proposed method can achieve high success ratio of object search even under situations with short query intervals.

4.2.4 Effect of Object Popularity

In real P2P systems, the popularity of objects is likely to follow a Zipf [20] distribution [21]. In a Zipf distribution, the probability p_i that a request for an i–th rank object occurs is expressed as follows:

$$p_i = \frac{i^{-\alpha}}{\sum_{k=1}^{n} k^{-\alpha}} \tag{1}$$

where n is the number of objects and α is a control parameter. In realistic cases, α is nearly one. In the following simulation, we set a unique rank to each object randomly

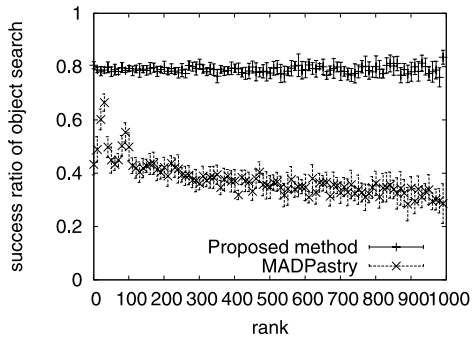

Fig. 9 Success ratio of object search at every ten rank under object popularity following a Zipf distribution.

selected from 1 to 1000. The node speed and the object size were set to 5.0 m/s and 8 bytes, respectively. The other parameters were the same as those in Sect. 3.

Figure 9 depicts the average success ratio of object search every ten rank with the 95% confidence interval. The success ratio of object search deteriorates as the rank becomes lower in MADPastry. On the contrary, it does not deteriorate in the proposed method. Although the results for high-rank objects in MADPastry oscillate, they will be stable by increasing the number of simulation runs.

In what follows, we discuss why these results occur. The following two problems occur in a P2P system over a MANET when the object popularity follows a Zipf distribution.

(i) For high-rank objects, the success ratio of object search deteriorates because concentration of queries to their object holders increases packet collisions.
(ii) For low-rank objects, the success ratio of object search deteriorates because of the route nonexistence or route disappearances.

The clustering method of MADPastry can absorb the second problem, especially the route disappearances in the routing between clusters, because a cluster has both high-rank and low-rank objects and queries for the low-rank objects can use routes established by the search for the high-rank objects in the same cluster. In addition, the proposed method can reduce the first problem and the part of the second problem, that is route nonexistence and route disappearances in routing inside a cluster, because a nodes can answer queries for any cluster member.

4.2.5 Effect of the Number of Clusters

The number of clusters determines redundancy of objects that affects the success ratio of object search. The smaller the number of clusters, the larger the redundancy. However, reducing the number of clusters also increases the overall traffic because the number of messages generated by broadcast inside a cluster is proportional to a square of the number of cluster members. In the above evaluations, we used 16 as the number of clusters as in Ref. [7]. In this section, we investigate the system performance by changing the number

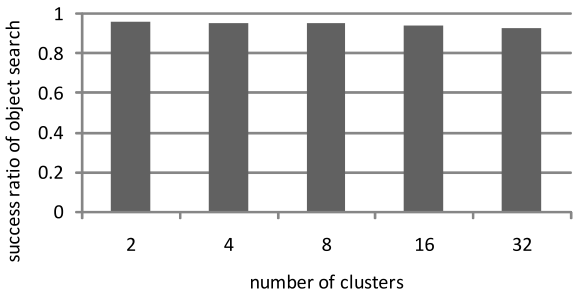

Fig. 10 Success ratio of object search when changing the number of clusters.

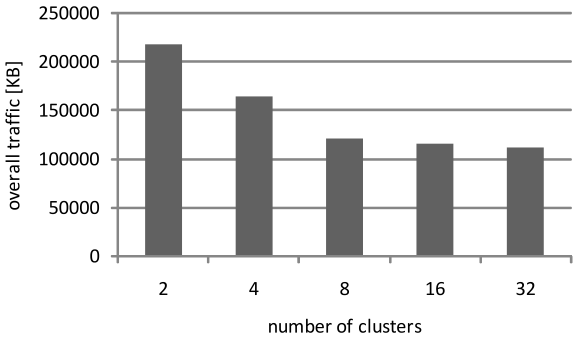

Fig. 11 Overall traffic when changing the number of clusters.

of clusters as 2, 4, 8, 16, and 32. The node speed is set to 1.4 m/s. The other parameters were the same as those in section 3.

Figures 10 and 11 demonstrate that the success ratio of object search and the overall traffic of the proposed method when the number of clusters is changed. Changing the number of clusters does not almost affect the success ratio of object search. On the other hand, the overall traffic monotonically decreases and converges to a certain value with an increase of the number of clusters.

4.2.6 Effect of Node Density

The node density also affects the system performance. If the node density is low, collisions at the network layer decrease but the network connectivity deteriorates. In what follows, we reveal the feasible area of the proposed method and broadcast in terms of the node density. Figures 12 and 13 demonstrate that the success ratio of object search and the overall traffic of the proposed method and broadcast when the node density (nodes/Km^2) is set to 62.5, 75, and 100. We find that the success ratio of object search slightly decreases with a decrease with the node density in both methods. On the other hand, the collisions at the network layer (the deterioration of network connectivity) mainly but slightly affect(s) the sift in the overall traffic in case of broadcast (the proposed method).

If the node density further deteriorates, the connectivity of the MANET falls below one due to the occurrence of nodes without neighboring nodes and fragmentation of the network. In such situations, the proposed method cannot

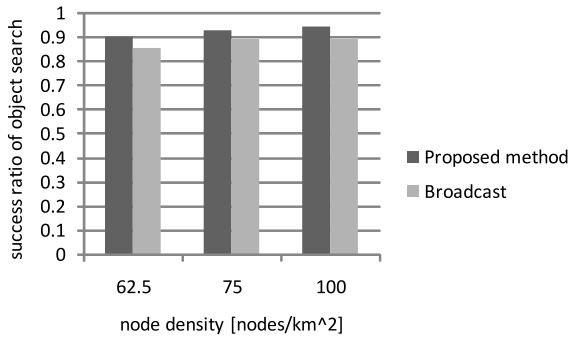

Fig. 12 Success ratio of object search when changing node density.

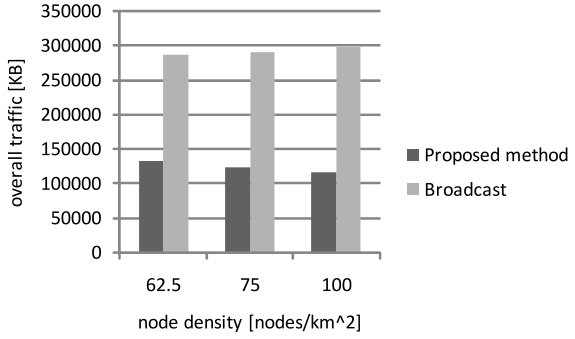

Fig. 13 Overall traffic when changing node density.

construct an overlay network and depresses the success ratio of object search. However, such networks with poor connectivity can be regarded as one of delay/disruption/disconnect tolerant networks (DTNs) in which the conventional store-forward routing, including broadcast and the proposed method, cannot work well [22]. Although store-carry-forward routing has been studied in DTNs, such situations are outside the scope of this paper.

5. Conclusion and Future Work

In this paper, we first explained the problems of MADPastry under high-mobility environments: lots of packet losses occur due to link disconnections at the network layer. We extended MADPastry by adding a method to share objects among cluster members to make the system robust to the node mobility. Through simulation experiments, we showed that the proposed method could achieve up to 2.5 times larger success rate of object search than MADPastry.

In addition, we investigated how the following parameters affected the system performance: the object size, the query interval, the popularity of objects, the number of clusters, and the node density. The proposed method is suitable for text-based information sharing in which the object size is less than 1000 bytes. The query interval affects the traffic load on the system. A short query interval may cause heavy congestion while a long query interval may induces lots of AODV route disappearances. However, the proposed method could work well at the moderate query interval. We also showed that the proposed method can achieve high success ratio of object search independently of the object popularity. The appropriate number of clusters was 16 to achieve high success ratio of object search with low overall traffic. The proposed method can achieve high system performance if the node density was over 62.5 nodes/Km^2.

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. In addition, we plan to formulate a new metric to model the network dynamics and extend the proposed method by taking into account the metric. In this paper, we focused on only the node velocities as a metric that represents the degree of network dynamics. However, the network dynamics should be modeled by the relationship among node speed, transmission range, and node density.

Acknowledgement

We would like to appreciate Post Doctor T. Zahn and Prof. J. Schiller who provided us with the source code of MADPastry.

This research was supported by a Grant-in-Aid for Young Scientists (B) 17700058 in Japan.

References

[1] D. Johnson, D. Maltz, and J. Broch, DSR: The dynamic source routing protocol for multihop wireless ad hoc networks, ch. 5, pp.139–172, Addison-Wesley, 2001.
[2] C.E. Perkins and E.M. Royer, "Ad-hoc on-demand distance vector routing," Proc. Second IEEE Workshop on Mobile Computer Systems and Applications, pp.90–100, IEEE Computer Society, Feb. 1999.
[3] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, Oct. 2003.
[4] "Gnutella." avairable at http://www.gnutella.com/
[5] M. Conti, E. Gregori, and G. Turi, "A cross-layer optimization of gnutella for mobile ad hoc networks," Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp.343–354, May 2005.
[6] H. Pucha, S.M. Das, and Y.C. Hu, "Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks," Proc. Sixth IEEE Workshop on Mobile Computing Systems and Applications, pp.163–173, Dec. 2004.
[7] T. Zahn and J. Schiller, "MADPastry: A DHT substrate for practicably sized MANETs," Proc. Fifth Workshop on Applications and Services in Wireless Networks, June 2005.
[8] M. Caesar, M. Castro, E.B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: Network routing inspired by DHTs," Proc. 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp.351–362, Sept. 2006.
[9] H. Pucha, S.M. Das, and Y.C. Hu, "Ekta+: Opportunistic multiplexing in a wireless DHT," Proc. First International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking, pp.69–71, July 2006.
[10] F. Delmastro, "From pastry to CrossROAD: CROSS-layer ring overlay for ad hoc networks," Proc. Third IEEE International Conference

on Pervasive Computing and Communications Workshops, pp.60–64, March 2005.
[11] A.I.T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object address, and routing for large-scale peer-to-peer systems," Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), pp.329–350, Nov. 2001.
[12] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," Proc. 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp.149–160, Aug. 2001.
[13] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area address and routing," Tech. Rep. UCB/CSD-01-1141, EECS Department, University of California, Berkeley, April 2001.
[14] M.K. Marina and S.R. Das, "Ad hoc on-demand multipath distance vector routing," SIGMOBILE Mob. Comput. Commun. Rev., vol.6, no.3, pp.92–93, July 2002.
[15] S.J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," Proc. IEEE Wireless Communications and Networking Conference (WCNC 2000), pp.23–28, Sept. 2000.
[16] "The network simulator ns-2." avairable at http://www.isi.edu/nsnam/ns/
[17] 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, Aug. 2002.
[18] J. Aspnes and G. Shah, "Skip graphs," Proc. Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.384–393, Jan. 2003.
[19] C. Schmidt and M. Parashar, "Enabling flexible queries with guarantees in P2P systems," IEEE Internet Comput., vol.8, no.3, pp.19–26, 2004.
[20] G.K. Zipf, Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology, Addison-Wesley, 1949.
[21] K. Sripanidkulchai, "The popularity of GnutellaQueries and its implications on scalability," White Paper, avairable at http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html, 2001.
[22] S. Farrell and V. Cahill, Delay- and Disruption-Tolerant Networking, Artech House, Norwood, MA, USA, 2006.

Kei Takeshita received the M.E. degree from Osaka University, Osaka, Japan, in 2008. He will be assigned as a researcher of NTT laboratories, Tokyo, Japan. His research interest includes mobile P2P networking and human interface.

Masahiro Sasabe received the M.E. and Ph.D. degrees from Osaka University, Osaka, Japan, in 2003 and 2006, respectively. He is currently an Assistant Professor with the Department of Information and Communication Technology, Osaka University. From 2003 to 2004, he was a Research Fellow with 21COE-JSPS, Japan. From 2004 to 2007, he was an Assistant Professor with the Cybermedia Center, Osaka University. His research interests include QoS architecture for multimedia distribution system, P2P communications, and ubiquitous networking. Dr. Sasabe is a member of the Institute of Electrical and Electronics Engineers.

Hirotaka Nakano received the B.E., M.E., and D.E. degrees in electrical engineering from Tokyo University, Tokyo, Japan, in 1972, 1974, and 1977, respectively. From 1999 to 2004, he served as the head in the Multimedia Laboratory, NTT DOCOMO, where he is currently a Professor with the Cybermedia Center, Osaka University, Osaka, Japan. In 1977, he joined NTT Laboratories, Tokyo, Japan, where he was engaged in research and development of videotex systems and multimediaon- demand systems. From 1995 to 1999, he was an Executive Manager of the Multimedia Systems Laboratory, NTT Human Interface Laboratories. His research areas include ubiquitous networks. Dr. Nakano is a member of the Institute of Image Information and Television Engineers of Japan, and the Institute of Electrical and Electronics Engineers.