*K*NN Query Processing Methods in Mobile Ad Hoc Networks

Yuka Komai, Yuya Sasaki, Takahiro Hara, *Senior Member, IEEE,* and Shojiro Nishio, *Fellow, IEEE*

Abstract—In this paper, we propose two beacon-less *k*NN query processing methods for reducing traffic and maintaining high accuracy of the query result in mobile ad hoc networks (MANETs). In these methods, the query-issuing node first forwards a *k*NN query using geo-routing to the nearest node from the point specified by the query (query point). Then, the nearest node from the query point forwards the query to other nodes close to the query point, and each node receiving the query replies with the information on itself. In this process, we adopt two different approaches: the Explosion (EXP) method and the Spiral (SPI) method. In the EXP method, the nearest node from the query point floods the query to nodes within a specific circular region, and each node receiving the query replies with information on itself. In the SPI method, the nearest node from the query point forwards the query to other nodes in a spiral manner, and the node that collects a satisfactory *k*NN result transmits the result to the query-issuing node. Experimental results show that our proposed methods reduce traffic and achieve high accuracy of the query result, in comparison with existing methods.

Index Terms—MANETs, *k*NN query, locations-based service

1 Introduction

Location-based service (LBS) is a typical application for *mobile ad hoc networks* (*MANETs*) [4,11]. In an LBS, it is common for a node to issue *k* Nearest Neighbor (*k*NN) queries [2,10,16,18], which search the information on the *k* nearest neighbors (*k*NNs) from the specified location (query point).

Many in-network processing techniques for *k*NN queries have been proposed for wireless sensor networks (WSNs). For example, in [3,13,14], the authors proposed infrastructure-free *k*NN query processing methods, which are efficient in retrieving *k*NN information with low message overhead (traffic) since they do not require the maintenance of network infrastructures. In these methods, nodes must exchange messages including node information (e.g., 1-hop beacon messages including the location of the sender).

However, MANETs possess notable characteristics, such as limitations on network bandwidth, and dynamic topology change due to the movement of mobile nodes. When nodes frequently exchange beacon messages to accurately know changing locations of neighboring nodes, traffic increases, and this causes frequent packet losses, resulting decrease in the accuracy of the query result. On the other hand, when nodes do not frequently exchange beacon messages, traffic is reduced, but nodes cannot accurately know the neighboring nodes' locations.

Therefore, in MANETs it is desirable to retrieve *k*NN information without using beacons. In this case, since nodes do not acquire the information on neighboring nodes beforehand, *k*NN query processing requires on-demand acquisition of neighboring nodes' information. A naive approach achieving this involves the query-issuing node flooding a *k*NN query over the entire network, and every node receiving the query sending back a reply. However, in this approach, many unnecessary replies, not included in the result of the *k*NN query (*k*NN result), are sent back to the query-issuing node, and thus traffic increases. Moreover, in MANETs, existing approaches that construct a static logical network (e.g., a query propagation tree) do not work. This is because when the network topology changes during query execution, the query-issuing node may not be able to acquire all the information on *k*NNs.

Fig. 1 shows an example of performing a *k*NN query, where an event organizer distributes some coupons to *k*NNs from a specific location. If the query-issuing node receives replies from all nodes, it produces a large amount of unnecessary traffic. Moreover, if a static query propagation tree is constructed and a link in the tree is disconnected, the query-issuing node cannot acquire the information on some *k*NNs.

In this paper, we propose two beacon-less *k*NN query processing methods for reducing traffic and maintaining high accuracy of the query result in MANETs. In these methods, the query-issuing node first forwards a *k*NN query using geo-routing to the nearest node from the query point. Then, the nearest node from the query point forwards the query to other nodes close to the query point, and each node receiving the query replies with the information on itself. In this process, we adopt two different approaches: the Explosion (EXP) method and the Spiral (SPI) method. Since these methods do not require flooding of a *k*NN query

• *The authors are with the Deptartment of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Osaka 565-0871, Japan.*
E-mail: {komai.yuka, sasaki.yuya, hara, nishio}@ist.osaka-u.ac.jp.

Manuscript received 7 May 2012; revised 16 July 2013; accepted 3 Oct. 2013. Date of publication 17 Oct. 2013; date of current version 15 May 2014.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, *and reference the Digital Object Identifier below.*
Digital Object Identifier 10.1109/TMC.2013.133

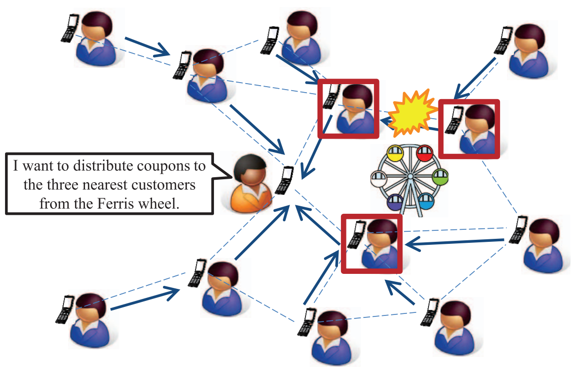

I want to distribute coupons to the three nearest customers from the Ferris wheel.

Fig. 1. Example of a *k*NN query in a MANET.

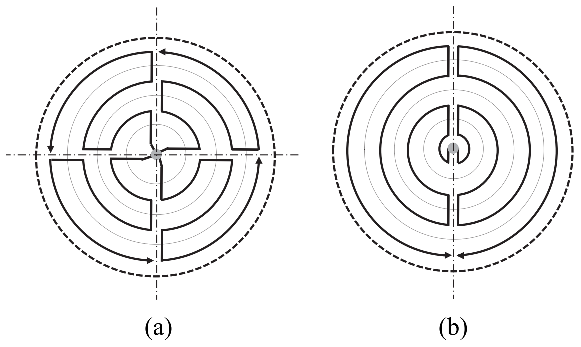


Fig. 2. Itinerary structures.

over the entire network, beacon messages between nodes, or network infrastructure, they can solve problems faced by the naive and existing approaches, caused by bandwidth limitations and dynamic network topology change.

The contributions of this paper are as follows:

- In MANETs, it is highly important to minimize unnecessary query messages and replies (i.e., traffic). In this paper, we propose effective *k*NN query processing methods for reducing traffic while preserving high accuracy of the query result.
- We propose two different methods, the Explosion (EXP) method and the Spiral (SPI) method. The performance of these methods is affected by several factors, such as the number of nodes (node density), movement speed, and *k*.
- Through extensive simulations, we show that our methods can reduce traffic compared with the naive method and the beacon-based (periodic message exchanges) method, and also achieve high accuracy of the query result. In addition, we take a clue to adaptively choose one of the EXP and SPI methods under various system situations.

The remainder of this paper is organized as follows. In Section 2, we introduce related work. In Section 3, we present our proposed *k*NN query processing methods. In Section 4, we discuss the results of the simulation experiments, and in Section 5, we summarize the paper.

2 Related Work

Many strategies have been proposed for processing *k*NN queries in WSNs [6,8,17]. In our paper, we focus on some of the most relevant existing studies on WSNs involving features similar to MANETs, such as wireless communication and multi-hop relaying, and discuss the differences in our approach.

In [3,13,14], the authors proposed *k*NN query processing methods in location-aware sensor networks based on the query propagation method proposed in [15]. In [13], the authors proposed an infrastructure-free *k*NN query processing method called DIKNN. This method consists of three phases; routing phase, *k*NN boundary estimation phase, and query dissemination phase. In this method, first a sink sends a query message to the nearest node from the query point. The information on the density of nodes is collected during the query transmission. The nearest node to the query point estimates the *k*NN boundary (which represents a search range), based on the information on the density of nodes, and partitions the *k*NN boundary into sectors. In each sector, a sensor node collects partial results (the information on nodes within its communication range), and propagates the query to the next node in the sector according to a well-devised itinerary structure (Fig. 2(a)). Finally, the partial results are individually sent back from each sector to the query-issuing node, enabling the query-issuing node to acquire *k*NNs. In [3], the authors proposed an infrastructure-free *k*NN query processing method called PCIKNN which consists of phases same as DIKNN. This method also sets the *k*NN boundary and partitions it into some sectors. With respect to each sector, a sensor node collects partial results along a well-devised itinerary structure, and then, the last node in each sector sends back the information on the nodes in each sector to the nearest node from the query point. After the nearest node from the query point receives the information on nodes in all sectors, it aggregates the partial results and sends them back to the query-issuing node. In DIKNN and PCIKNN, by partitioning the search range, the response time of query execution can be reduced even if the search range is large.

In [14], the authors proposed three methods for processing *k*NN queries in location-aware sensor networks: the GRT, KBT and IKNN algorithms. In the GRT and KBT algorithms, a tree infrastructure composed of sensor nodes is constructed, and a *k*NN query is propagated along it. The IKNN algorithm is an infrastructure-free approach, which propagates a query along a calculated spiral route (Fig. 2(b)). The SPI method proposed in this paper is inspired from the IKNN algorithm.

However, these all algorithms basically assume a static or location-aware sensor network. More specifically, each sensor node must precisely know its neighbors (e.g., by frequently exchanging beacon messages), which causes excessive overhead in highly dynamic MANETs.

As described later, to process a *k*NN query without beacons, a node must process it on the fly using only the information included in a query and its location information. In addition, it is necessary to search the entire range where *k*NNs are present, and to identify the conclusion of a search based solely on the information in a query. Our proposed methods are more suitable to the MANET environments, because we have designed methods to meet these requirements.

3 *K*NN Query Processing Methods

In this section, we first describe the assumed environment, and the design policy of our methods. Then, we present how *k*NN queries are processed in our methods.

3.1 Assumptions

The system environment is assumed to be a MANET in which all mobile nodes have the same radio communication facility, whose communication range is a circle of a fixed size. We assume that the MANET is sufficiently dense that network partitioning does not occur, and geo-routing can be performed between any pair of nodes. In the MANET, mobile nodes retrieve the information on other mobile nodes by using *k*NN queries. The query-issuing node transmits a query message associated with the query point, and acquires the information on the *k* nearest nodes from the query point among all nodes in the entire network.

We assign a unique *node identifier* to each mobile node in the system. The set of all mobile nodes in the system is denoted by $M = \{M_1, M_2, \cdots, M_n\}$, where n is the total number of mobile nodes and M_i $(1 \leq i \leq n)$ is a node identifier. Each mobile node moves freely.

Every mobile node knows the total number of mobile nodes (n), the size of the entire area in which mobile nodes are present, and its current location (using a positioning system such as GPS).

3.2 Design Policy

The *k* nearest nodes from the query point in a MANET change during a search, because nodes move freely. Therefore, the query-issuing node must acquire the query result in a short time. To this end, we have designed our methods to execute queries using only one round of message transmissions.

In MANETs, it is highly important to minimize the amount of information transmitted, due to limitations on network bandwidth and the batteries of the mobile nodes. For example, packet losses and packet retransmissions may occur when the network is congested, resulting in information failing to be transmitted. In this sense, the periodic broadcast of beacon messages by nodes, even when no node is searching *k*NNs in the network, causes unnecessary traffic; and thus a beacon-less method is more suitable for MANETs. However, without exchanging beacon messages, nodes cannot know neighboring nodes' information beforehand. Therefore, we need to design beacon-less methods involving on-demand searches. In doing so, we should avoid using message flooding over the entire network, because it is particularly wasteful use of network bandwidth.

In addition, a node must process a query in a distributed manner on the fly using only the information included in the query and its location information; and nodes must cooperate with neighbors to process a query with low traffic.

3.3 Overview of Our Methods

As mentioned above, it is important to process a *k*NN query without beacons and message flooding over the entire network. To process a *k*NN query without beacons, nodes must

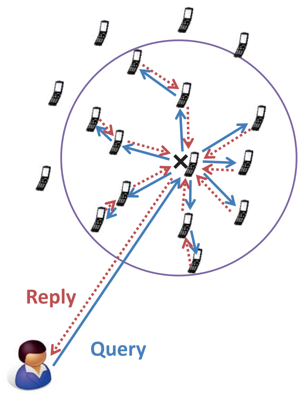


Fig. 3. EXP method.

employ on-demand acquisition of information on neighbors, in order to forward a query message. In our methods, we adopt a three-way handshake transmission to forward a message. Specifically, the node closest to the query point transmits a reply message after the shortest waiting time, and other nodes stop sending replies upon overhearing this reply, in order to reduce the traffic.

Using geo-routing based on this message forwarding manner, in our methods, first the query-issuing node transmits a *k*NN query to the nearest node from the query point (the *global coordinator*). Then, the global coordinator commences the process of acquiring the information on *k*NNs. In this process, we adopt two different approaches, the Explosion (EXP) and Spiral (SPI) methods. In both methods, nodes cooperate with neighbors to process a query using only the information included in the query and its location information.

In the EXP method (Fig. 3), the global coordinator floods the *k*NN query to nodes within a specific circular region centered on the query point, resembling an 'explosion' of the query message from the global coordinator. The size of the circular region is determined based on the density of nodes in the entire area. Each node receiving the query replies to the global coordinator with the information on itself. After collecting replies from the nodes in the circular region, the global coordinator replies, with the *k*NN result, to the query-issuing node.

In the EXP method, each node receiving a query message replies when the waiting time, which is based on the distance between the query point and its location, has passed, and the farther nodes set a lesser waiting time. By doing so, the intermediate nodes can aggregate replies from farther nodes.

The SPI method (Fig. 4) does not require specifying the specific region for sending a query message unlike the EXP method. In the SPI method, the entire area is dynamically partitioned into a set of hexagonal cells whose size is determined based on the communication range of the mobile nodes, with the query point at the center of a hexagonal cell. The global coordinator begins forwarding the *k*NN query, in a spiral manner, to the nodes located nearest the central point of the respective hexagons (*local coordinator*). Each local coordinator collects the information on all nodes in its hexagonal cell, and then forwards the query, including this collected information, to the next hexagon in the spiral. The

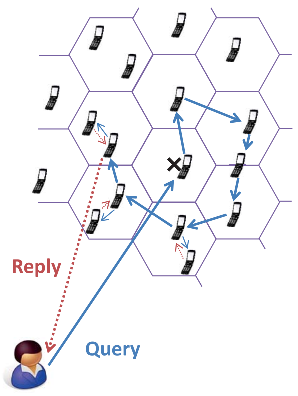


Fig. 4. SPI method.

node that surely collects the *k*NN result transmits the result to the query-issuing node.

In the SPI method, each node is aware of the hexagon to which it belongs, solely based on the information in the query (i.e., the query point) and its location. The local coordinator can both acquire the information on nodes within the same hexagon, and determine the next node to which to transmit the query, using the same query.

In both methods, a designated node aggregates the information in the received replies, and thus unnecessary information is not sent in reply through a long path to the query-issuing node. In these ways, unnecessary transmissions of queries and replies can be reduced.

3.4 Geo-Routing for Forwarding a Query

To dynamically construct a route to the query point, we extend a beacon-less geo-routing method proposed in [5]. Our geo-routing method adopts a three-way handshake protocol to enable a given node to find the node nearest to the query point among its neighboring nodes, and then send a query to this node. By repeating this procedure, the query is forwarded to the global coordinator.

Specifically, in our geo-routing method, first the query-issuing node broadcasts a neighbor search message. When a node receives this message, if it is closer to the query point than the source node, and within the *searching range* (discussed below), it stores the ID of the source node as the parent node and sets the waiting time, *Reply_Delay* (*RD*), for sending a reply, according to the following equation:

$$RD = Max\_delay \cdot \left(\frac{\alpha - d}{\alpha}\right), \tag{1}$$

where *Max_delay* is a positive constant specifying the maximum waiting time before sending a reply, α is the radius of the searching range, and d is the distance between the source node's location and the foot of the receiving node's perpendicular to the line from the source node to the query point. The searching range is a circular region centered on the source node, and must always be lesser than the communication range. This is because if it is set as equal to the communication range, there will be frequent link disconnections between nodes and their parents during query processing. Specifically, it is determined based on the speed only of the source node (without factoring in the speed of other nodes), because it is too costly to acquire information on the other nodes' mobility. The maximum distance that the node moves during a query transmission ($Xmax$) is calculated as $Xmax = Vmax \cdot T$, where $Vmax$ is the maximum speed of the node, and T is the search time. Thus, if the radius of the searching range is equal to the communication range minus $Xmax$, the query path is probably available during a query transmission. However, our method still work well even if α is roughly defined. This is because the node can return the results by using geo-routing once again even if a link disconnection between the node and its parent occurs.

According to Equation (1), nodes closer to the query point transmit reply messages after shorter waiting times. Thus, the node with the minimum *RD* is the first to transmit a reply message to the source node of the neighbor search message. The other nodes that received this reply stop sending a reply if they set the *RD*. The source node of the neighbor search message, having received reply messages from its neighbors, sends a (forwards the) *k*NN query message only to the node that first sent the reply. The node that receives the forwarded *k*NN query broadcasts a neighbor search message. Finally, if the node that sends a neighbor search message does not receive any reply message when the query point is within its communication range, it recognizes itself as the global coordinator and begins acquiring *k*NNs. In this way, the query-issuing node can forward a *k*NN query to the global coordinator with little traffic, because the geo-routing method requires neither beacon messages nor the construction of multi-paths.

During the execution of the geo-routing method, each mobile node can recognize its parent. Therefore, nodes can utilize the constructed path to forward a query for the reply. Because a node's parent is selected from nodes within the searching range, link disconnections between nodes and their parents rarely occur.

Fig. 5 shows an example of forwarding a query message using our geo-routing method, when M_1 issues a *k*NN query. M_1 transmits a neighbor search message to its neighboring mobile nodes. On receiving the message, M_2 and M_3 respectively set their *RD*. Since M_4 is not within the searching range, it ignores the message. Since M_5 is farther from the query point than M_1, it also ignores the message. M_2, with the minimum *RD*, is the first to transmit a reply message to M_1, when its *RD* has passed. M_1, on receiving the reply message from M_2, forwards the *k*NN query to M_2. M_3, which received the reply message from M_2, makes no reply to M_1. Then, M_2 transmits a neighbor search message in the same manner as M_1. Finally, M_6, which is the nearest to the query point (the global coordinator), receives the *k*NN query.

3.5 Forwarding a *k*NN Query and Returning the Result : Explosion (EXP) Method

3.5.1 Concept of EXP method

There must be *k*NNs within the circle centered on the query point and whose radius is decided based on the estimated distance between the query point and the *k*-th nearest node from the query point. However, it is difficult to accurately determine the optimal radius of this circle, because nodes

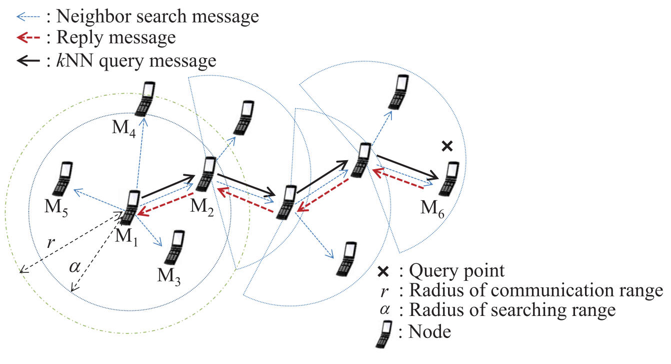

: Neighbor search message
: Reply message
: kNN query message
M4
M5
M2
M1
M3
M6
r
α
× : Query point
r : Radius of communication range
α : Radius of searching range
: Node

Fig. 5. Example of executing the geo-routing method.

do not know the locations of other nodes beforehand. If the query-issuing node liberally estimates the radius of the circle, it can acquire the accurate *k*NN result with high probability. However, if every node within this expansive circle replies with its information to the query-issuing node, unnecessary replies (from nodes that are in the circle but are not *k*NNs) are sent back through long paths to the query-issuing node.

To solve this problem, in our methods the global coordinator first collects the information on all nodes within the circle, and then sends the aggregated result to the query-issuing node. By doing so, a query and its replies are transmitted with small hopcounts, which helps to reduce the traffic. Here, in order for nodes in the circle to efficiently aggregate the node information, individual nodes should not send replies separately, but should relay replies from nodes farther from the query point to closer nodes. However, since nodes do not know other nodes' locations, it is difficult to achieve this. Thus, our idea is to set waiting times before replying, based on the distance from the respective nodes to the global coordinator. Specifically, by setting shorter waiting times for nodes further from the global coordinator, closer nodes have more chance to relay or overhear and aggregate others' replies.

3.5.2 Query Processing

The behavior of the query-issuing node, M_s, and of mobile nodes receiving the query message, is as follows.

1) M_s specifies the requested number of *k*NNs, k, and the query point. Then M_s determines the radius of the estimated *k*NN circle, R, based on the entire area size, the total number of nodes in the area, and k, by the following equation:

$$R = \sqrt{\frac{k \cdot area}{\pi \cdot n}}, \tag{2}$$

where n is the total number of nodes in the entire network, and *area* is the area size.

2) M_s transmits a *k*NN query message to the nearest node from the global coordinator, using the geo-routing method described in Section 3.4. In the query message, the query-issuing node's ID and location are set as M_s and its location, respectively, the requested number of *k*NN is set as k, the radius of the estimated *k*NN circle is set as R, and the query point is set as the location specified by the query.

3) By the process described in Section 3.4, the global coordinator, M_p, is selected. Then, M_p broadcasts a local query message to its neighboring mobile nodes. In the message, the requested number of *k*NNs is set as k, the radius of the estimated *k*NN circle is set as R, the query point is set as that in the received query message, and the global coordinator's ID and location are set as M_p and its location, respectively.

4) Each mobile node, M_q, that initially receives the local query message stores the identifier of the source node, M_p, as its *EXP parent*. If M_q is within the estimated *k*NN circle, it sets the waiting time, *Wait Time* (*WT*), for sending a reply, according the following equation:

$$WT = \beta \cdot \left(\frac{R}{r}\right) \cdot \left(1 - \frac{a}{R+b}\right), \tag{3}$$

where a is the distance between M_p and M_q, b is the distance between the query point and M_p, β is a parameter specified by a system designer to avoid message collision, and r is the communication range. As Equation (3) shows, *WT* decreases as the distance between M_p and M_q increases.
At the same time (without waiting *WT*), M_q broadcasts a local query message to its neighboring mobile nodes.
If M_q has already received a local query message, or is not within the estimated *k*NN circle, it discards the message and does nothing.

5) The node that has set the least *WT* begins to transmit a reply message (after waiting *WT*) with the information on itself, including its location, to its EXP parent. This attached information is called the *tentative kNN result*.

6) Each node that receives the reply message (from its EXP child) updates the tentative *k*NN result included in the reply message, by adding the information on itself. If the number of nodes whose information is included in the tentative *k*NN result exceeds k, the information on the node farthest from the query point is removed from the tentative *k*NN result.
When *WT* has passed, if the node is not the global coordinator, it transmits a reply message, including the updated tentative *k*NN result, to its EXP parent, and the procedure returns to Step 5. If the respective node is the global coordinator, the procedure goes to Step 7.

7) When *WT* has passed, M_p replies, with the updated tentative *k*NN result as the final *k*NN result, to the query-issuing node, along the same path through which the query message was sent from the query-issuing node to M_p. If a link disconnection occurs along the path, the node that detects the disconnection transmits the reply using the geo-routing method described in Section 3.4, where the query point is set as the location of the query-issuing node.

Fig. 6 shows an example of executing the EXP method, where M_1 is the global coordinator; and Fig. 7 shows a flow diagram of the query processing. When M_5 receives a local

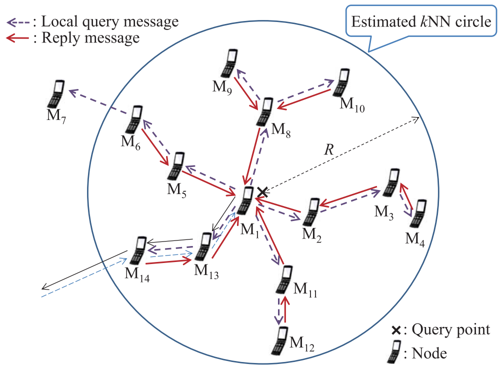


Fig. 6. Query processing in the EXP method.

query message from M_1, it stores M_1's ID as its EXP parent, and broadcasts a local query message to its neighboring nodes because it is within the estimated *k*NN circle. In the same way, on receiving the query message, M_6 stores M_5's ID as its EXP parent, and broadcasts a local query message to its neighboring nodes. In contrast, M_7, on receiving the query message, discards the message because it is not within the estimated *k*NN circle. When *WT* has passed at M_6, M_6 transmits a reply message, including M_6's information, to M_5. On receiving this message, M_5 transmits a reply message, including both its and M_6's information, when *WT* has passed. Such procedures are performed throughout the entire MANET, until finally M_1 acquires the information on all nodes within the estimated *k*NN circle. When *WT* has passed at M_1, it transmits the *k*NN result to the query-issuing node.

The EXP method can reduce the traffic required for collecting the *k*NN result, because the tree structure is dynamically constructed during transmissions of local query messages (by changing the waiting time based on the nodes' positions), and the information of nodes only in a specific region (the estimated *k*NN circle) is efficiently collected along the tree. However, in this method, the value of *R* must be appropriately set. If *R* is set too low, the information on some *k*NNs may not be acquired; while if *R* is set too high, the traffic increases unnecessarily due to the transmission of needless information. Our approach to determining *R* is based on the density of nodes in the entire MANET. However, in a real environment, it is not always easy to know the total number of nodes in the entire MANET, and the area size, in advance. Moreover, the density of nodes is generally not uniform in a MANET.

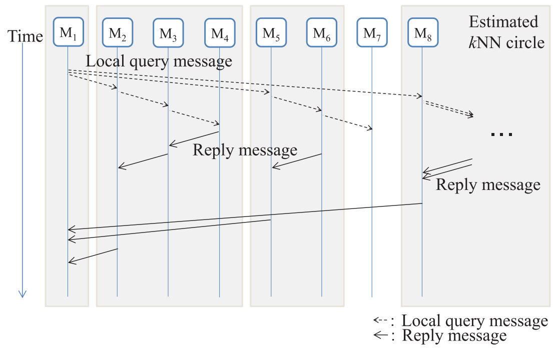


Fig. 7. Flow diagram of query processing in the EXP method.

3.6 Forwarding a *k*NN Query and Returning the Result : Spiral (SPI) Method

3.6.1 Concept of SPI method

As noted in Section 3.5, it is difficult to accurately determine the optimal (minimum) radius of the circle which includes *k*NNs. Therefore, a method to obtain *k*NNs without specifying the circle is needed. For this aim, we adopt the strategy of visiting nodes in a spiral manner following the IKNN algorithm in [14]. In this way, since a query travels along a single path, it can efficiently aggregate nodes' information (i.e., can filter out unnecessary information), and can halt the process at an appropriate time. We explain reasons in detail why we choose single path transmission in a spiral pattern.

(i) Initial stipulation of a search range is not needed.

In the existing methods including the EXP method, in which the search range is determined in advance based on the local information, misestimation of the search range degrades the performance. In contrast, since spiral propagation gradually extends the range without initial stipulation of the search range, it involves no risk of such misestimation.

(ii) Queries can search nodes in order of increasing distance from the node's location to the query point.

An efficient approach involves initially searching for nodes near the query point, and then gradually extending the range to acquire information on farther nodes. This avoids unnecessary extension of the search range. A spiral propagation pattern is suitable for this approach. The existing methods [3,13,14] also utilize the spiral propagation pattern.

(iii) The node which acquires kNNs appropriately initiates a reply to the query-issuing node.

In the SPI method, the node that (i) acquires the information on at least *k* nodes and (ii) ensures that the search so far has fully covered a circular range containing *k* nodes centered on the query point, can initiate a reply to the query-issuing node. In the existing methods [3,13,14], the query is replicated and multiple queries are processed in parallel to reduce the response time. However, it is difficult to judge when the search is over, because queries cannot know how many nodes other queries have searched so far. In contrast, spiral propagation presents no such difficulty.

(iv) Radio interference can be suppressed.

In the SPI method, *k*NNs are acquired by a single path in order to avoid radio interference; whereas, in the existing methods, multiple queries are processed in parallel, and the accuracy of the query result may decrease due to the resulting radio interference.

3.6.2 Partition of the Area

According to the above policy, in the SPI method, the entire area is partitioned into a set of hexagonal cells and the *k*NN query is forwarded in a spiral manner through a series of hexagonal cells. Because the communication range has a basically circular shape, it is efficient to partition the area into hexagonal rather than triangular or quadrangular cells. More specifically, by appropriately setting the size of the hexagonal cell, a query can acquire the information of all

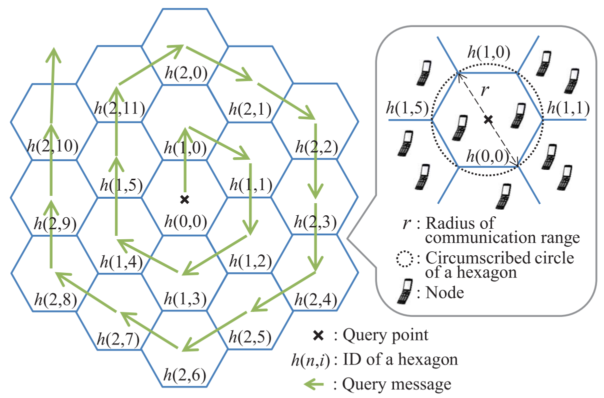


Fig. 8. Partition of the area.

nodes within each cell with one-hop message exchange, and then pass on to the next cell.

The entire area in which mobile nodes are present is uniformly divided into hexagonal cells so that the query point locates at the centroid of a cell. To guarantee that all mobile nodes in a hexagonal cell can receive messages sent by nodes in the same cell, we set the circumradius of each hexagonal cell as half the radius of the communication range. Here, as described in Section 3.1, we assume that each node knows the radius of the communication range, which is the same for all nodes. The hexagonal cells covering the entire area are uniquely determined. Each node autonomously determines to which hexagon it belongs, based on its location and the location of the query point included in the query message.

In the SPI method, when a node surely collects the information on kNNs, it transmits a reply message to the query-issuing node. Therefore, if there are kNNs within the circumscribed circle of the hexagonal cell which includes the query point, it is not necessary to transmit the kNN query to other hexagonal cells. If there are less than k nodes in the cell, the kNN query is transmitted (as necessary) to a series of concentric rings of hexagonal centered on the query point cell. Here, we define the hexagonal cell that includes the query point as the 0 lap hexagonal cell, the first concentric ring of hexagonal cells as the 1st lap (hexagonal cells), the second concentric circle as the 2nd lap, and so on. Each n-th lap hexagonal cell's ID is defined as $h(n,i)$ (i=0, $\cdots$, 6n-1), and i denotes the number given to the i-th hexagonal cell in a clockwise direction.

Fig. 8 shows an example of partitioning the area into a set of hexagonal cells and a typical query path through the structure, respectively. The center point of hexagonal cell $h(0,0)$ corresponds to the query point, and the dotted circle indicates the circumscribed circle of the corresponding hexagonal cell whose diameter equals the radius of the communication range. The 1st lap hexagonal cells with IDs $h(1,0)$ to $h(1,5)$ circumscribe to the 0 lap hexagonal cell $h(0,0)$. A query is forwarded from hexagonal cell $h(0,0)$, and travels through the 1st lap hexagonal cells, then the 2nd lap hexagon cells, in order.

3.6.3 Query Processing

In the SPI method, a kNN query is transmitted to a series of hexagonal cells (as described in Section 3.6.2). The behavior of the query-issuing node, M_s, and the mobile nodes receiving the query message is as follows.

1) M_s specifies the requested number of kNNs, k, and the query point, and transmits a kNN query to the global coordinator using the geo-routing method described in Section 3.4. In the query message, the query-issuing node's ID and location are set as M_s and its location, respectively, the requested number of kNNs is set as k, and the query point is set as the location specified by the user.
2) Through the process described in Section 3.4, the global coordinator, M_p, which locates within the 0 lap hexagon, is selected. M_p is called the *local coordinator* of the 0 lap region. M_p broadcasts a local query message to its neighboring mobile nodes. In the local query message, the query point is set as that in the received query message, the hexagon ID is set as $h(0,0)$, and the source node's location is set as M_p's location.
3) Each mobile node, M_t, which receives the local query message, transmits a reply message to the source node (local coordinator) if M_t is within the circumscribed circle of the hexagon corresponding to the hexagon ID in the received message. In the reply message, it includes the information on itself (its location and ID (M_t)).
4) M_t stores the query point and hexagon ID in the received message. Then, it calculates the center position of the hexagon which should be visited next. If M_t is closer to the center of the next hexagon than to that of the source node, it sets the waiting time (WT) for sending a *query request message* (described in Step 6) according to the following equation:

$$WT = Max\_delay \cdot \frac{h}{r} + \gamma, \tag{4}$$

where Max_delay is a positive constant specifying the maximum time before sending a query request message, h is the distance between M_t's location and the center of the hexagon which should be visited next, r is the radius of communication range, and γ is a margin of time for the local coordinator (source node) to receive replies from its neighbors. As Equation (4) shows, WT decreases with decreasing distance between M_t and the center of the next hexagon.

When WT has passed at M_t, it transmits a query request message, including its own ID, to the local coordinator.
5) The local coordinator receiving the reply messages from its neighboring nodes updates the *tentative kNN result* by adding the information in the received messages.

If the number of the information on nodes included in the tentative kNN result exceeds k, the information on nodes which are not kNNs from the query point is removed from the tentative result. Furthermore, if the local coordinator belongs to the last hexagon in the current lap, i.e., $h(n,5{\cdot}n)$ (n=1,$\cdots$), the procedure goes to Step 9.

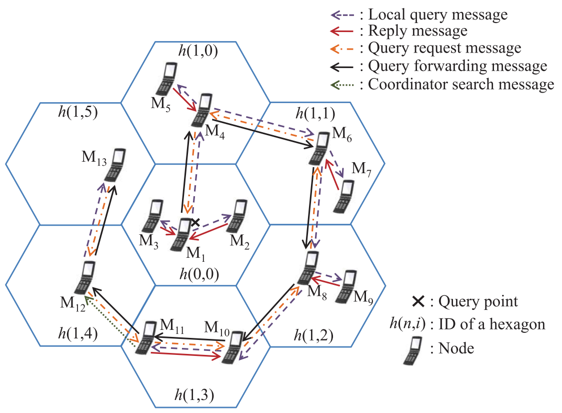


Fig. 9. Query processing in the SPI method.

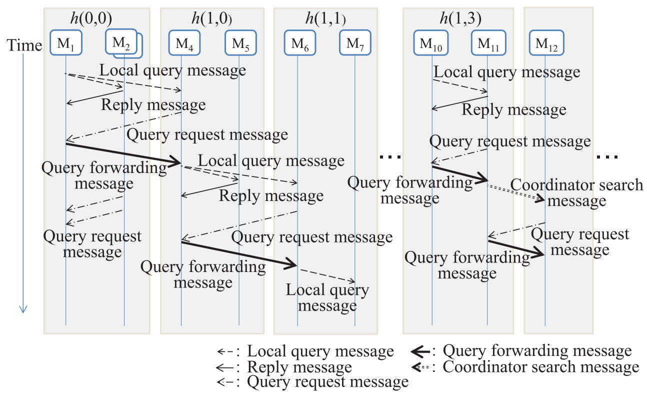


Fig. 10. Flow diagram of query processing in the SPI method.

6) When the local coordinator receives a query request message for the first time, it transmits a *query forwarding message* to the sender of the request message. This message includes the information on the requested number of *k*NNs, *k*, the query point, the query-issuing node's ID (M_s) and location, and the tentative *k*NN result. Meanwhile, the other nodes that receive the query request message stop sending query request messages.
7) The node that receives the query forwarding message becomes the new local coordinator. If it is within the hexagon which should be visited next, it broadcasts a local query message to its neighbor nodes, and the procedure returns to Step 3. If it is not within the next hexagon, it cannot collect the information in that hexagon, thus, it does not broadcast a local query message, but instead broadcasts a *coordinator search message* to its neighbor nodes.
8) Each mobile node, M_d, which receives the coordinator search message, stores the query point and hexagon ID in the received message. Then, it calculates the center positions of the hexagons which should be visited next and next but one. If M_d is (i) within the circumscribed circle of the next hexagon, or (ii) it is closer to the center of the next but one hexagon than that of the source node, it sets the waiting time (*WT*) before sending a query request message (described in Step 6) according to the following equation:

$$WT = \begin{cases} Max\_delay \cdot \frac{h_i}{r} & ((\text{i})), \\ Max\_delay \cdot (1 + \frac{h_{ii}}{r}) & \\ \qquad ((\text{ii}) \text{ without (i)}), \end{cases} \quad (5)$$

where *Max_delay* is a positive constant specifying the maximum time before sending a query request message, h_i is the distance between M_d's location and the center of the next hexagon to be visited, h_{ii} is the distance between M_d's location and the center of the next but one hexagon to be visited, and r is the radius of the communication range. As Equation (5) shows, *WT* becomes smaller as the distance between M_d and the center of the next hexagon decreases, and as the distance between M_d and the center of the next but one hexagon decreases. Here, if M_d does not meet the criteria, it discards the message.
When the *WT* has passed at M_d, the node transmits a query request message, including it's ID, to the local coordinator. The procedure goes to Step 6.
9) The local coordinator replies, with the tentative *k*NN result as the final result, to the query-issuing node using the geo-routing method described in Section 3.4, where the query point is set as the location of the query-issuing node. If the local coordinator or a relaying node incidentally knows a node on the query path from the query-issuing node to the global coordinator, it forwards the *k*NN result to this node, and the *k*NN result is sent back to the query-issuing node along the query path. If some nodes along the query path do not connect with their parents due to link disconnection, the local coordinator or relaying node again transmits the *k*NN result using the geo-routing method.

Fig. 9 shows an example of executing the SPI method, where M_1 is the global coordinator, and Fig. 10 shows a flow diagram of the query processing. First, M_1 broadcasts a local query message to its neighboring nodes. When M_2 and M_3 receive a local query message from M_1, they reply with their information to M_1 as a local reply because they are within the same hexagon as M_1. On receiving the local replies from M_2 and M_3, M_1 stores the information on itself, M_2, and M_3 as the tentative *k*NN result. M_4, on receiving the query message from M_1, transmits a query request message to M_1 when *WT* has passed. When M_1 receives this message from M_4, M_1 transmits a query message with the information on M_2, M_3, and itself. Such procedures are performed in all hexagons on the 1st lap, and finally M_{13} acquires the information on *k*NNs and initiates a reply to the query-issuing node.

The SPI method expands the searching area until a node surely collects the information on *k*NNs. Therefore, it is not necessary to set the searching area (i.e., *R* in the EXP method) in advance. We use "surely" when a query acquires the information on equal to or more than *k* nodes, and searches the whole circle area centered on the query point, and whose radius is the distance between the query point and the *k*-th farthest node from the query point among all searched nodes. Actually, the query-issuing node should be able to acquire the information on correct *k*NNs in most cases. However, due to message collisions, the query-issuing node may not always find *k*NNs.

Here, the SPI method can address situations in which no node is present in a hexagon through the transmission of a coordinator search message. If there is no node in a given hexagon, a query continues searching to bypass an empty hexagon by passing through neighboring nodes as a detour. Specifically, if M_w (which sent a local query message) receives no query request message (e.g., there is no candidate in the next hexagon among the neighbors), after a given time it sends a coordinator search message to find the next local coordinator. After that, each node follows the regular procedures. Thus, the SPI method can resolve the problem. If, due to a network partition, M_w does not receive the message from any nodes by 2.5·*Max_delay* after sending the coordinator search message, a query discontinues searching.

The SPI method is typically time-consuming to forward a query, and thus significant response time is involved when a query must traverse a large number of hexagons. In such a situation, the network topology may change during the execution of this method, which results in changes in the *k*NNs. Therefore, it may be difficult to maintain high accuracy of the query result.

3.7 Discussion

3.7.1 Node Mobility

Given the nodes' mobility in MANETs, it is more efficient to dynamically construct a query path than to construct a static path in advance, and our methods involve such dynamic construction. Specifically, in our geo-routing method, which uses three-way handshakes without exchanging beacon messages, nodes select the next hop using the searching range, taking into account node mobility. Thus, a link disconnection between a given node and its parent does not occur very often; and if such a disconnection occurs, the node can reply (with the *k*NNs) to the query-issuing node by using geo-routing once again. In our simulation experiments, the graphs regarding node speed (Fig. 13(c)) show that our methods can maintain high accuracy of the query result, even when nodes move fast (up to 20.0[m/s], i.e., 72[km/h], which implies that some nodes are high-speed vehicles).

3.7.2 Appearance and Disappearance of Nodes

In this paper, we do not incorporate the appearance and disappearance of nodes during query processing. If a query-relaying node disappears during query processing, the acquired result may be lost. However, such a situation can be easily avoided by assigning a neighbor as an alternative coordinator and passing the result to this neighbor. We will design the concrete algorithm in our future work. We should also note that such appearance and disappearance rarely occurs during query processing in our methods, because query processing (response time) is quite brief (roughly a few seconds).

3.7.3 Estimated *k*NN Circle in the EXP method

The EXP method determines the search range (estimated *k*NN circle) as it simply supposes that nodes are uniformly distributed while it is not always true in a real environment. Even if nodes are not evenly distributed, the EXP method is executable (can work). Of course, in such a situation, the estimated *k*NN circle is not very accurate, and thus, its performance may degrade.

Specifically, if the estimated *k*NN circle in the EXP method contains less than *k* nodes due to misestimation, the accuracy of the query result decreases because the information on the remaining *k*NNs is not resought. However, the EXP method can easily adapt to this situation. When it acquired less than *k* nodes, the global coordinator could replace the circle with a larger one, and search again. Another approach to adapt to a network whose node density is not uniform is to estimate the density of nodes near the query point before spreading a query (e.g., by exchanging messages between neighbors). However, this causes an increase in the traffic and the delay; thus, we need an alternative, more effective and cost-efficient means to estimate node density. Setting the estimated *k*NN circle appropriately in such a situation is a challenging research issue, which is part of our future work.

Many applications require high accuracy of the query result. However, even if the accuracy of the query result does not reach 1, some applications, such as the distribution of coupons or advertisements in a city or event site, are still suitable. In the simulation experiments in this paper, we use the "MAP value" to measure the accuracy of the query result, because the answers with a higher rank are more important than those with a lower one in a *k*NN search. In particular, when the accuracy of the query result does not reach 1, the answers generally do not include the information on nodes with a low rank, due to node movement and inhomogeneous density. Moreover, query processing by repetition, or a large estimated *k*NN circle, with the aim of achieving perfect accuracy of the query result, performs poorly in terms of traffic and delay. In addition, it is difficult to achieve perfect accuracy of the query result because nodes move freely, which causes variations in distance between node locations and the query point (i.e., the *k*NNs dynamically change). That is why we aim to balance between accuracy and traffic, rather than achieving perfect accuracy with too much (unacceptable) overhead.

4 Simulation Experiments

In this section, we discuss the results of simulation experiments evaluating the performance of our proposed methods. For the simulation experiments, we used the network simulator, QualNet4.0[12].

4.1 Simulation Model

The number of mobile nodes in the entire system is n ($M_1, \ldots, M_n$). These mobile nodes are present in an area of 1,000[m] × 1,000[m], and move according to the random waypoint model [1], with a movement speed and pause time of from 0.5 to v[m/sec] and 3[sec], respectively. Note that we also conducted simulations with other mobility models: the random walk and random waypoint models with a home area. In the latter model, the entire area is partitioned into four square regions of equal size, and each node selects its next destination either from the region in which it resides (90% probability) or from another region (10%). The results show that our proposed methods achieve roughly the same performance in all the three mobility models, and the differences in performance between our methods and comparative methods are almost same in the three mobility models. Thus, we only show here the results with the random waypoint model.

Mobile nodes transmit messages using an IEEE 802.11b device with a data transmission rate of 11[Mbps]. The transmission power of the mobile nodes is determined such that the radio communication range is about 100[m]. Packet losses and delays occur due to radio interference. We assume that each node knows its current location. The query point specified by a *k*NN query is (500, 500). The *Max_delay* in Equations (1) and (4) is set at 0.1 based on our preliminary experiments, so that our methods achieve good performance. α in Equation (1), β in Equation (3), and γ in Equation (4) are respectively set at 97, $n/500$, and $0.005 \cdot n/30$, based on our preliminary experiments.

We compare the performance of our proposed EXP and SPI methods with that of a naive method, DIKNN[13], and the EXP and SPI methods using beacons. In the naive method, which does not use beacons, the query-issuing node broadcasts a *k*NN query to its neighbor nodes, and then the query is transmitted to nodes within a specific region using the geo-routing method proposed in [7]. Specifically, each mobile node that receives the *k*NN query re-broadcasts the query to its neighbor nodes if it is nearer to the query point than the sender node, or is within the estimated *k*NN circle (which is the same as that in the EXP method). In addition, if a node that receives the *k*NN query is within the estimated *k*NN circle, it directly replies with the information on itself, to the query-issuing node, along the path through which the query message was sent. In the DIKNN method, the search range is set by the method proposed in [13], and partitioned into four sectors in these simulation experiments. In the EXP and SPI methods using beacons, the node behavior is basically the same as in the beacon-less EXP and SPI methods, however messages are transmitted (unicast) based on the information obtained by beacon messages (i.e., nodes' IDs and locations). In the methods using beacons (DIKNN, and the EXP and SPI methods using beacons), the broadcasting period of beacon messages is set at 20[sec] by default, based on our preliminary experiments[1]. In Section 4.4, we evaluate two different cases, with beacon periods of 10 and 20[sec] respectively.

The requested number of *k*NNs, *k*, is varied from 1 to 50, with nodes' maximum movement speed, *v*, set at 1.0[m/s]. The total number of nodes, *n*, is set at 500 in Section 4.2 and 250 in Section 4.3. In addition, to evaluate the impact of node mobility, *v* is varied from 0.5 to 20[m/s] in Section 4.4, where *k* and *n* are respectively set at 10 and 500.

In the above simulation model, we randomly determine the initial position of each mobile node. An hour later, the query-issuing node, randomly chosen among all nodes, issues a *k*NN query. We repeat this process 1000 times (i.e., 1000 queries) for every 20 seconds, and evaluate the following three criteria.

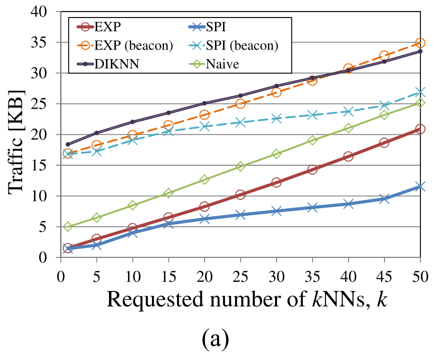
1. If nodes exchange messages more frequently than every 20 sec in DIKNN, they can acquire more accurate information on their neighbors. However, the traffic increases, which causes packet losses due to message collision. We have selected 20 sec as the beacon period, because good DIKNN performance is achieved in terms of the traffic and the accuracy of the query result.

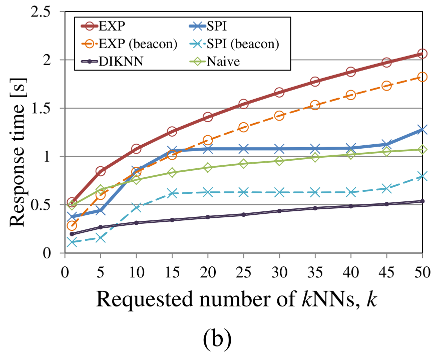
TABLE 1
Message Types and Sizes

Type	Size [B]
Query (naive method)	48
Reply (naive method)	24
Neighbor search (geo-routing)	48
Reply (geo-routing)	8
Query (EXP method)	56
Query (SPI method)	48
Query (DIKNN method)	$56{+}24{\cdot}l$
Local query (EXP, SPI and DIKNN methods)	56
Coordinator search (SPI method)	24
Query request (SPI method)	8
Query forwarding (SPI method)	$72{+}16{\cdot}i$
Reply (EXP, SPI and DIKNN methods)	$32{+}16{\cdot}i$
Ack to the received reply	8
Beacon	32

- Traffic
  We examine the total volume of query messages and replies exchanged in processing a query. Table 1 shows the size of messages used, in our methods and the comparative methods, where *i* denotes the number of nodes whose information is included in the reply, and *l* denotes the number of hops to the query-issuing node. We define "traffic" as the average of the total volume of respective queries issued.
  Here, energy evaluation is highly important in MANET environments. Energy consumption is primarily constituted by computational calculations and message transmissions. In our approaches, message transmissions predominate in the search for *k*NNs, and calculation costs are very small. Thus, we consider that the evaluation of energy consumption is roughly the same as that of the traffic.
- Response time
  We examine the time from the transmission of a query message by the query-issuing node, to the reception of receiving the *k*NN result. In the naive method, the response time is defined as the time from the query-issuing node's transmission of a query message, to its reception of the last reply message. (Note that, in a real environment, the query-issuing node cannot recognize which reply is the last one.) We define "response time" as the average of such times for all queries issued.
- Accuracy of query result
  We examine the ratio of the number of *k*NNs whose information is included in the *k*NN result acquired by the query-issuing node, to the requested number of *k*NNs, *k*. We define "accuracy of query result" as the MAP (Mean Average Precision) value, which measures the performance of the result with a ranking [9]. The MAP is an average of the AP (Average Precision) of the respective queries. The AP and MAP are determined by the following equations:

$$AP_i = \frac{1}{k}\sum_{j=1}^{k}\frac{g}{j}\cdot e, \qquad (6)$$





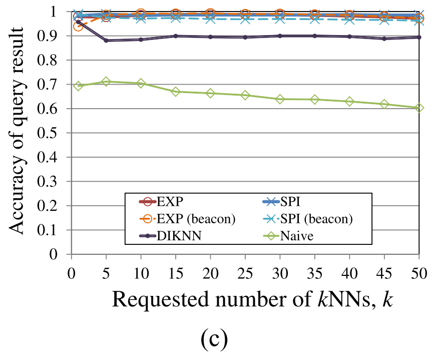


Fig. 11. Case of 500 nodes. (a) Traffic. (b) Response time. (c) Accuracy of query result.

$$MAP = \frac{1}{querynum} \sum_{i=1}^{querynum} AP_i, \tag{7}$$

where AP_i is the AP of the i-th issued query, g is the number of nodes which are included in the query result among the top-j nearest nodes, $querynum$ is the total number of issued queries (i.e., 1,000 in this simulation), and e is determined by the following equation:

$$e = \begin{cases} 1 \text{ (j-th nearest node is included} \\ \quad \text{in the kNN result).} \\ 0 \text{ (otherwise).} \end{cases} \tag{8}$$

Thus, the MAP increases as the query-issuing node acquires the information on nodes nearer to the query point.

4.2 Simulation Results in the Case of 500 Nodes

First, we examine the performance of our proposed methods when the number of nodes is 500. Fig. 11 show the simulation results. In these graphs, the horizontal axis indicates the requested number of kNNs, k, and the vertical axis indicates the traffic in Fig. 11(a), the response time in Fig. 11(b), and the accuracy of query result in Fig. 11(c).

From Fig. 11(a), as k increases, the traffic increases in all methods, because both the search area for processing a kNN query and the data volume of the reply increase. Our proposed methods generate far less traffic than the methods using beacons, as the periodical beacon exchanges involved in the latter cause a great deal of traffic. In particular, the DIKNN method produces the largest traffic in most cases, because it generates many unnecessary replies from every partitioned sector. Our proposed methods also generate far less traffic than the naive method, because in the former, a kNN query is first forwarded to the nearest node from the query point, with a small hopcount and without multiple paths, using the geo-routing method. The EXP method produces more traffic than the SPI method, because in our simulations the specified radius of the estimated kNN circle is relatively large for safety, and thus many non-kNN nodes reply. In the SPI method, the traffic depends on the number of laps required for collecting the information on kNNs, and thus the traffic increases in a stepwise manner as k increases.

From Fig. 11(b), the response time in our proposed methods is greater than in the methods using beacons. By using the information on neighbor nodes obtained from beacon messages, a node can forward a message without a three-way handshake or waiting time for reply. Furthermore, because the DIKNN method can acquire the information on nodes in parallel, by simultaneously searching all partitioned sectors, the response time is very short. In our methods, on the other hand, a node sets a waiting time before transmitting a message via geo-routing (i.e., RD), and thus the response time is increased. In the EXP method in particular, every node must wait for WT before sending back a reply, which results in an increase in response time. In the naive method, such waiting times do not occur; however, in this method retransmissions of replies often occur, due to packet losses caused by the increased traffic. Overall, the response time in the naive method is roughly similar to that of the SPI method.

From Fig. 11(c), the accuracy of query result is very high (nearly 1) in our methods, in contrast to the lower accuracy of query result in the DIKNN and naive methods. This is because, in the latter, packet losses often occur due to individual replies, from all sectors in the DIKNN method, and from all nodes in the naive method.

4.3 Simulation Results in the Case of 250 Nodes

Next, we examine the performance of our proposed methods when the number of nodes is 250 (low node density). Fig. 12 show the simulation results. In these graphs, the horizontal axis indicates the requested number of kNNs, k, and the vertical axis indicates the traffic in Fig. 12(a), the response time in Fig. 12(b), and the accuracy of query result in Fig. 12(c).

From Fig. 12(a), in the methods using beacons, the traffic is less than in the case of 500 nodes, owing to the decreased traffic in beacon message exchange. The traffic in the naive method is also less than in the case of 500 nodes, because the traffic involved in propagating a kNN query decreases. In the EXP method, the traffic is roughly the same as in the case of 500 nodes, because in the geo-routing employed by our methods, a kNN query can be transmitted to the nearest node from the query point with a small hopcount, regardless of the node density, and the estimated kNN circle is determined based on the node density. In the SPI method, on the other hand, the traffic is greater than in the case of 500 nodes, because the method requires more laps in order to collect the information on kNNs when the density is lower.

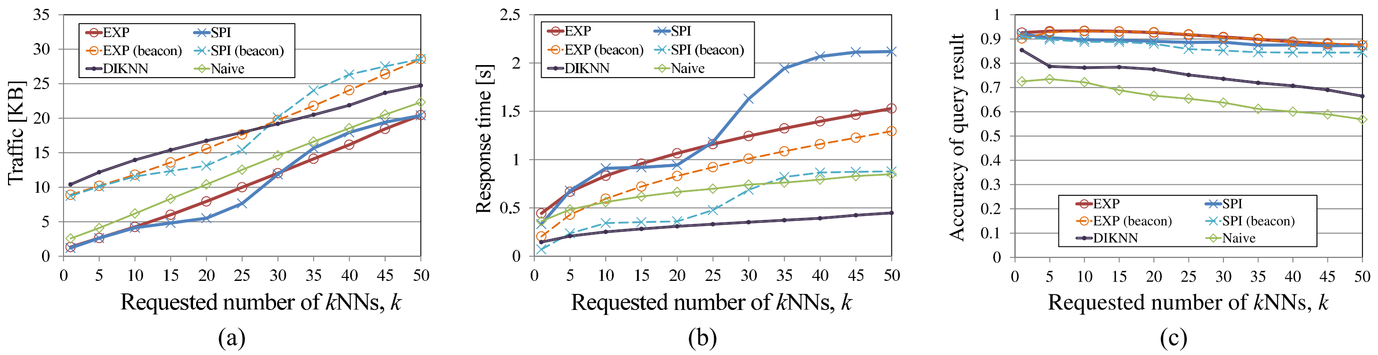


Fig. 12. Case of 250 nodes. (a) Traffic. (b) Response time. (c) Accuracy of query result.

From Fig. 12(b), the response time in the SPI method is much greater than in the case of 500 nodes due to the increase in laps, as described above. In the other methods, the response time is less than in the case of 500 nodes, because some of the parameters determining waiting time were based on node density (lessened for low density) in order to avoid message collision.

From Fig. 12(c), no methods can achieve 100% accuracy of the query result, because with low node density it often happens that there is no available node for the next hop when transmitting a query and reply during geo-routing. In the EXP method, the accuracy of the query result is slightly higher than in the other methods, because a query message is propagated to nodes near the query point by flooding, and thus the occasions for using geo-routing are less than in the other methods.

4.4 Impact of Node Mobility

Finally, we examine the performance of our proposed methods by varying node speed when the number of nodes is 500. Fig. 13 show the simulation results. In these graphs, the horizontal axis indicates the maximum speed of nodes, v, and the vertical axis indicates the traffic in Fig. 13(a), the response time in Fig. 13(b), and the accuracy of query result in Fig. 13(c); "(beacon xs)" denotes a method using beacons, with a period of x seconds. In these simulation experiments, k is set at 10.

From Fig. 13(a) and 13(b), the traffic and response time are nearly constant, regardless of the nodes' speed. This shows that link disconnections rarely occur during the execution of all the methods except for the DIKNN method, because in the former the execution time is short. In the DIKNN method, as v increases, the response time slightly increases, because as nodes move faster, the chance of a link disconnection increases during the transmission of replies from all sectors.

From Fig. 13(c), the accuracy of the query result in our proposed methods remains high, regardless of the nodes' speed, because the path of message transmission is reactively constructed. In methods using beacons, on the other hand, as v increases, the accuracy of the query result significantly decreases when the period of the beacon message is set at 20[s]. This shows that nodes cannot accurately know neighboring nodes' information, due to the mobility of nodes. While setting the period of the beacon message at a lower value, such as 10[s], can maintain high accuracy of the query result, it generates significantly greater traffic, as shown in Fig. 13(a). Thus, we can confirm that the methods using beacons are not suitable for a highly dynamic network.

4.5 Discussion

4.5.1 Comparison Among Six Methods

The respective methods have advantages in different applications or network conditions. Table 2 compares the features of the respective methods.

In the methods employing beacons ("yes" at Beacon column in Table 2), the frequency of the broadcasting period of beacon messages directly affects the performance of the methods. If nodes frequently exchange beacon messages, *k*NNs can be searched with high accuracy of the query result, because nodes can accurately know the information on their neighbors. However, this involves much traffic, which often makes the accuracy of the query result of the query result worse than beacon-less methods. In terms of

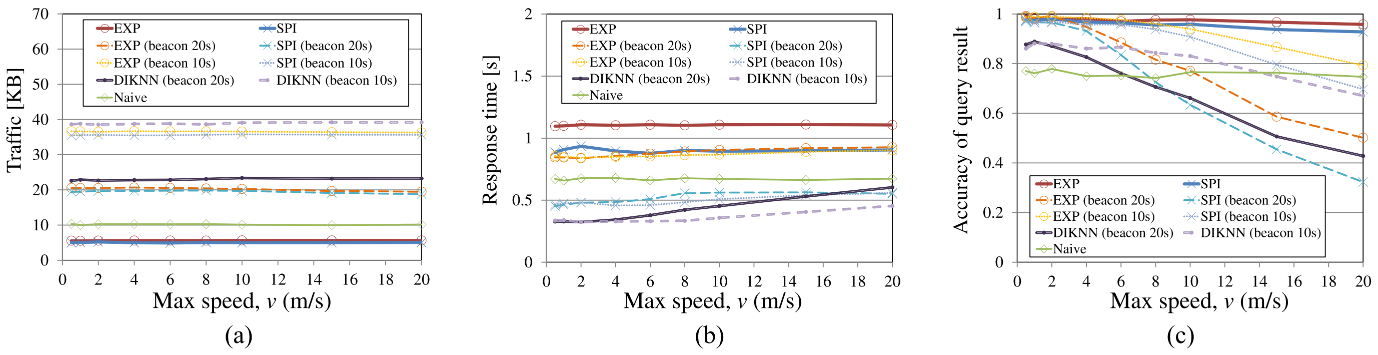


Fig. 13. Case of varying node speed. (a) Traffic. (b) Response time. (c) Accuracy of query result.

TABLE 2
Comparison Table

Method	Beacon	Next hop decision		Search
		to query point	kNN collection	Range
EXP	no	Inquiry	(Broadcast)	Proactive
SPI	no	Inquiry	Inquiry	Reactive
EXP(beacon)	yes	Beacon info.	(Broadcast)	Proactive
SPI(beacon)	yes	Beacon info.	Beacon info.	Reactive
DIKNN	yes	Beacon info.	Beacon info.	Proactive
Naive	no	(Broadcast)	-	Proactive

response time, a node which wants to forward a query can immediately determine the next hop node, which results in a short response time. Therefore, methods employing beacons are useful in networks where nodes already exchange beacon messages frequently in processing other applications, which means that no additional beacon costs are required. For example, such methods are suitable for collaborative work (e.g., a rescue operation in a disaster) in which nodes often frequently exchange messages to collaborate with other members, and which typically requires high accuracy of the query result. The EXP and SPI methods using beacons are suitable for such a situation. On the other hand, the response time of DIKNN is shorter than that of the other methods, owing to the use of multiple paths, but DIKNN often cannot achieve high accuracy of the query result. DIKNN is thus suitable for applications that do not required highly accurate results, but require that the results be acquired as soon as possible (e.g., the distribution of time-service information in an event site).

Beacon-less methods ("no" at Beacon column in Table 2) can work well in networks where nodes do not know the information on neighbors. In particular, these methods can maintain the accuracy of the query result even if nodes move quickly. Moreover, in MANETs, a situation often occurs in which some nodes are not available to frequently exchange beacon messages, due to limitations in network bandwidth. Beacon-less methods typically work well in such situations. The response time of the (beacon-less) EXP and SPI methods is larger than that of methods with beacons because a node determines the next hop node using a three-way handshake. However, in the real environment, many applications tolerate the delay of a few seconds for searching kNNs. For example, in an application which distributes event information in a city, our beacon-less methods are suitable for acquiring kNNs because these methods do not involve much network consumption.

In the EXP method and DIKNN, the search range which includes kNNs with high probability is determined in advance ("Proactive" at Search Range column in Table 2). To estimate the search range appropriately, the estimated node density must be almost equal to the real node density within the search range. However, when the node density is strongly skewed throughout the entire network, it is hard to accurately estimate the search range, and thus the accuracy of the query result decreases or traffic increases. Therefore, these methods work well when the estimation of the search range is reliable (i.e., when either nodes know the configuration of the node density, or the node density is uniform in the network and known). For example, in a rescue operation in a disaster, in which members often know the allocation of members in regions, the query-issuing node can know the number of members near a query point. On the other hand, in the SPI method, since a query gradually extends the search range by acquiring kNNs, an initial estimation of the search range is not needed. A suitable application here, for example, would be coupon distribution in a city, where it is often difficult for each node to know the number of nodes in the area.

4.5.2 How to Choose Two Proposed Methods

We proposed two different methods, EXP and SPI, and their performance superiority (which is better) depends on some environmental factors such as density of nodes. In the following, we briefly explain our current idea of how to adaptively choose on appropriate method based on the density of nodes in a real environment.

In this paper, we assume that each node knows the total number of nodes and the size of the entire area where nodes are present. Therefore, each node can calculate the average density of nodes in the entire area using this information. The query-issuing node can select either the SPI or EXP method based on the average node density; the former method if the average density is high, and latter if it is low. However, in a real environment, it is not always easy to know such information in advance. In a naive way, when a node can count the number of nodes in the entire network and estimate the area size by flooding a message to all nodes and receiving replies from them. However, this is not efficient, because flooding involves significant traffic.

Therefore, we need a method to estimate the density of nodes without flooding. For example, a node would broadcast a message to its neighbors, receive replies, and thereby roughly estimate the local density of nodes. After such estimation, the query-issuing node selects the EXP method or SPI method. In future, we will consider concrete cost-efficient means to estimate node density.

5 Conclusion

In this paper, we have proposed kNN query methods for reducing traffic and maintaining high accuracy of the query result in MANETs. In our methods, the query-issuing node first forwards a kNN query using geo-routing to the nearest node from the query point. Then, the nearest node from the query point forwards the query to other nodes close to the query point, and each node that receives the query replies with the information on itself. In this process, we adopt two different approaches: the Explosion (EXP) and Spiral (SPI) methods.

The experimental results show that our proposed methods both reduce the traffic required for processing kNN queries, in comparison with naive and existing methods using beacons, and achieve high accuracy of the query result. The EXP method can achieve both reduction in traffic and high accuracy of the query result at any levels of node density; while the SPI method can achieve high performance in an environment with high node density, though it does not perform well in a sparse environment. It should be noted that since nodes locate at random positions in our experiments, there is basically no skew in node distribution over the entire area. This is advantageous to the EXP method, which determines the radius of the estimated *k*NN circle, *R*, based on the density of nodes over the entire area. Performance evaluation in an environment with skewed node distribution will figure in our future work.

ACKNOWLEDGMENTS

This research is partially supported by the Grant-in-Aid for Scientific Research (S)(21220002) and (B)(24300037) of MEXT, Japan.

REFERENCES

[1] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, 2002.

[2] C.-Y. Chow, M. F. Mokbel, and H. V. Leong, "On efficient and scalable support of continuous queries in mobile peer-to-peer environments," *IEEE Trans. Mobile Comput.*, vol. 10, no. 10, pp. 1473–1487, Oct. 2011.

[3] T.-Y. Fu, W.-C. Peng, and W.-C. Lee, "Parallelizing itinerary-based KNN query processing in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 711–729, May 2010.

[4] T. Hara and S. K. Madria, "Consistency management strategies for data replication in mobile ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 7, pp. 950–967, Jul. 2009.

[5] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Walchli, "BLR: Beacon-less routing algorithm for mobile ad-hoc networks," *Comput. Commun.*, vol. 27, no. 11, pp. 1076–1086, 2004.

[6] P. P. Jayaraman, A. Zaslavsky, and J. Delsing, "Cost-efficient data collection approach using K-nearest neighbors in a 3D sensor network," in *Proc. MDM*, Kansas City, MO, USA, 2010, pp. 183–188.

[7] Y.-B. Ko and N. H. Vaidya, "Flooding-based geocasting protocols for mobile ad hoc networks," *Mobile Netw. Applicat.*, vol. 7, no. 6, pp. 471–480, 2002.

[8] W.-C. Lee and B. Zheng, "DSI: A fully distributed spatial index for location-based wireless broadcast services," in *Proc. ICDCS*, Columbus, OH, USA, 2005, pp. 349–358.

[9] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[10] T. P. Nghiem, A. B. Waluyo, and D. Taniar, "A pure peer-to-peer approach for kNN query processing in mobile ad hoc networks," *Pers. Ubiquit. Comput.*, vol. 17, no. 5, pp. 973–985, Jun. 2013.

[11] C. E. Perkins and E. M. Royer, "Ad hoc on demand distance vector routing," in *Proc. WMCSA*, New Orleans, LA, USA, 1999, pp. 90–100.

[12] *Scalable Network Technologies: "Qualnet"* [Online]. Available: http://www.scalable-networks.com/

[13] S.-H. Wu, K.-T. Chuang, C.-M. Chen, and M.-S. Chen, "Toward the optimal itinerary-based KNN query processing in mobile sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1655–1668, Dec. 2008.

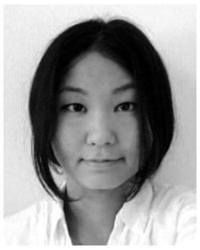
[14] Y. Xu, T.-Y. Fu, W.-C. Lee, and J. Winter, "Processing k nearest neighbor queries in location-aware sensor networks," *Signal Process.*, vol. 87, no. 12, pp. 2861–2881, 2007.

[15] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "Processing window queries in wireless sensor networks," in *Proc. ICDE*, 2006, p. 70.

[16] B. Xu, F. Vafaee, and O. Wolfson, "In-network query processing in mobile P2P databases," in *Proc. GIS*, Seattle, WA, USA, 2009, pp. 207–216.

[17] Y. Yao, X. Tang, and E.-P. Lim, "Localized monitoring of kNN queries in wireless sensor networks," *VLDB J.*, vol. 18, no. 1, pp. 99–117, 2009.

[18] X. Yu, K. Q. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving object," in *Proc. ICDE*, 2005, pp. 631–642.



Yuka Komai received the B.E. degree in multimedia engineering and the M.E. degree in information science and technology from Osaka University, Osaka, Japan, in 2011 and 2013, respectively. Currently, she is a Ph.D. candidate in information science and technology from Osaka University, Osaka, Japan. Her current research interests include distributed databases, mobile networks, and mobile computing systems.

Yuya Sasaki received the B.E. degree in multimedia engineering and the M.E. degree in information science and technology from Osaka University, Osaka, Japan, in 2009 and 2011, respectively. Currently, he is a Ph.D. candidate in Information Science and Technology from Osaka University, Osaka, Japan. His current research interests include data search and replication mechanisms in mobile computing environments.

Takahiro Hara received the B.E, M.E, and the Ph.D. degrees in information systems engineering from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is an Associate Professor in the Department of Multimedia Engineering, Osaka University, Osaka, Japan. He has published over 300 international Journal and conference papers in the areas of databases, mobile computing, peer-to-peer systems, WWW, and wireless networking. He served and is serving as a Program Chair of IEEE International Conferences on Mobile Data Management (MDM'06 and 10) and Advanced Information Networking and Applications (AINA'09 and 14), and IEEE International Symposium on Reliable Distributed Systems (SRDS'12). He guest edited *IEEE Journal on Selected Areas in Communications*, special issue on Peer-to-Peer Communications and Applications. His current research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a senior member of IEEE and ACM and a member of three other learnedsocieties.

Shojiro Nishio received the B.E., M.E., and the Ph.D. degrees from Kyoto University, Kyoto, Japan, in 1975, 1977, and 1980, respectively. He has been a Full Professor at Osaka University since August 1992, and is currently a Distinguished Professor of Osaka University. He served as a Vice President and Trustee of Osaka University from August 2007 to August 2011. He also acted as the Program Director in the Area of Information and Networking, Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan from April 2001 to March 2008. His current research interests include database systems and multimedia systems for advanced networks such as broadband networks and mobile computing environment. He has co-authored or co-edited over 55 books, and authored or co-authored more than 600 refereed journal or conference papers. He served as the Program Committee Co-Chairs for several international conferences including DOOD 1989, VLDB 1995, and IEEE ICDE 2005. He has also served as an editor of international journals including *IEEE Transactions on Knowledge and Data Engineering*, *VLDB Journal, ACM Transactions on Internet Technology*, and *Data & Knowledge Engineering*. He has received numerous awards during his research career, including the Medal with Purple Ribbon from the Japanese Emperor in 2011, which is awarded to people who have made outstanding and important contributions in academic fields, arts and sports. He is also a fellow of IEEE, IEICE, and IPSJ, and is a member of four learned societies, including ACM.