Research on Query Processing Methods for Location-based Services in MANETs

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2016

Yuka KOMAI

List of Publications

1. Journal Paper

1. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: KNN Query Processing Methods in Mobile Ad Hoc Networks, *IEEE Trans. on Mobile Computing*, Vol. 13, No. 5, pp. 1090–1103 (2014).

2. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: k-Nearest Neighbor Search Based on Node Density in MANETs, *Mobile Information Systems*, Vol. 10, No. 4, pp. 385–405 (2014).

3. Komai, Y., Nguyen, D.H., Hara, T., and Nishio, S.: kNN Search Algorithm with Index of the Minimum Road Travel Time in Time-Dependent Road Networks, *DBSJ Journal*, Vol. 13, No. 1, pp. 43–49 (2015).

4. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: k Nearest Neighbor Search for Location-Dependent Sensor Data in MANETs, *IEEE Access*, Vol. 3, No. 1, pp. 942–954 (2015).

2. International Conference Paper

1. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: A kNN Query Processing Method in Mobile Ad Hoc Networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 287–288 (2011).

2. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Searching k-Nearest Neighbor Nodes Based on Node Density in Ad Hoc Networks, in *Proc. of Int'l Conf. on Advances in Mobile Computing & Multimedia (MoMM)*, pp. 199–207 (2012).

3. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k Nearest Neighbor Queries for Location-Dependent Data in MANETs, in *Proc. of Int'l Conf. on Database and Expert Systems Applications (DEXA), Part II*, pp. 213–227 (2013).

4. Tsuda, T., Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Top-K Query Processing and Malicious Node Identification Against Data Replacement Attack in MANETs, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 279–288 (2014).

5. Komai, Y., Hara, T., and Nishio, S.: A Convex Hull Query Processing Method in MANETs, in *Proc. of Int'l Symposium on Reliable Distributed Systems (SRDS)*, pp. 331–332 (2014).

6. Komai, Y., Nguyen, D.H., Hara, T., and Nishio, S.: kNN Search Utilizing Index of the Minimum Road Travel Time in Time-Dependent Road Networks, in *Proc. of Int'l Workshop on Future Technologies for Smart Information Systems (FTSIS)*, pp. 131–137 (2014).

7. Komai, Y., Hara, T., and Nishio, S.: Processing Convex Hull Queries in MANETs, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 64–73 (2015).

8. Tsuda, T., Komai, Y., Hara, T., and Nishio, S.: Signature-Based Top-k Query Processing against Data Replacement Attacks in MANETs, in *Proc. of Int'l Symposium on Reliable Distributed Systems (SRDS)*, pp.130–139 (2015).

3. Domestic Conference Paper (with peer-review)

1. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k Nearest Neighbor Queries for Location-Dependent Data in Ad Hoc Networks (in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 17–25 (2012).

2. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k Nearest Neighbor Queries for Duplicated Location-Dependent Data in Ad Hoc Networks (in Japanese), in *Proc. of IPSJ DICOMO Symposium*, pp. 553–560 (2013).

3. Tsuda, T., Komai, Y., Hara, T., and Nishio, S.: Top-K Query Processing and Malicious Node Identification Against Data Replacement Attack in Ad Hoc Networks (in Japanese), in *Proc. of IPSJ DICOMO Symposium*, pp. 569–576 (2013).

4. Komai, Y., Nguyen, D.H., Hara, T., and Nishio, S.: kNN Search with Index of the Minimum Road Travel Time in Time-Dependent Road Networks (in Japanese), in *Proc. of IPSJ DICOMO Symposium*, pp. 1166–1173 (2013).

5. Komai, Y., Hara, T., and Nishio, S.: Convex Hull Query Processing Methods in MANETs (in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 191–198 (2014).

6. Kato, R., Hara, T., Shirakawa, M., Komai, Y., Majima, H., Amagata, D., Osawa, J., Matsuo, K., Yokoyama, M. Sazaki, H., Nakamura, T., Mizuno, S., Nishio, S.: Dummy-Based Anonymization System Implementation for Learning of Location Privacy (Demo Paper, in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 50–52 (2014).

4. Domestic Conference Paper

1. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Searching k-Nearest Neighbor Nodes in Ad Hoc Networks (in Japanese), in *Proc. of DEIM Forum*, online (2011).

2. Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Searching k-Nearest Neighbor Nodes Based on Node Density in MANETs (in Japanese), in *Proc. of DEIM Forum*, online (2012).

3. Tsuda, T., Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Top-k Query Processing and Malicious Node Identification based on Node Grouping in MANETs (in Japanese), in *Proc. of DEIM Forum*, online (2014).

4. Tsuda, T., Komai, Y., Hara, T., and Nishio, S.: Signature-Based Top-k Query Processing against Data Replacement Attacks in Ad Hoc Networks (in Japanese), in *Proc. of DEIM Forum*, online (2015).

Abstract

Recently, there has been an increasing interest in *mobile ad hoc networks (MANETs)*, which are comprised solely of mobile nodes. Since no special infrastructure is required, many applications are expected to be developed for MANETs, in various fields such as military affairs, rescue operations at disaster sites, and information sharing at event sites. Location-based services (LBS) are typical applications for MANETs, which are typically composed of a large number of nodes over a wide area. In an LBS, real-time location-specific queries to search for information held by mobile nodes are often used; and in such cases, it is effective to process the queries as k nearest neighbor (kNN) queries which acquire the information on kNNs from the specified location (*query point*), and convex hull queries which retrieve the information necessary for calculating the convex hull of all the nodes composing a given network. Although there has been many existing studies for kNN search and convex hull detection in networks where there is a central server, there is no research focused on query processing of such queries in MANETs.

MANETs possess notable characteristics, such as limitations on network bandwidth, and dynamic topology change due to the movement of mobile nodes. Therefore, a naïve approach acquiring the information on all nodes within the entire network does not work well because it produces excessive message transmission (*i.e.*, traffic), resulting decrease in the accuracy of the query result due to packet losses. In addition, existing location-specific query processing methods in wireless sensor networks do not work well either because in wireless sensor networks, sensor nodes are stationary or loction-aware (*i.e.*, know their own neighbors in advance). In MANETs, it is difficult to accurately know the information on neighboring nodes since the network topology dynamically changes due to the movement of mobile nodes. Therefore, we design query processing methods for reducing traffic and maintaining high accuracy of the query result without knowing neighboring nodes' information in advance.

Moreover, in an LBS, it is required to search for not only nodes themselves but also data items associated with a given location (location-dependent data). As nodes move in MANETs, data items which have been held by a node since long time ago are likely no longer related to the node's current location, because these data items are associated

with the node's former location. In such cases, the query-issuing node cannot effectively search for requested data items using location information. Therefore, new techniques searching for location-dependent data items are required.

In this thesis, we propose kNN and convex hull query processing methods for LBS in MANETs. This thesis consists of five chapters. First, we introduce the research background and issues for LBS in MANETs in Chapter 1. In Chapters 2 and 3, we address kNN query processing to search for the k nearest nodes and for the k nearest data items, respectively. In Chapter 4, we address query processing for calculating the convex hull of nodes in MANETs and also discuss how to know the convex hull of location-dependent data items in MANETs. Finally, in Chapter 5, we summarize this thesis and discuss our future work.

More specifically, in Chapter 2, we propose two kNN query processing methods to search k nearest nodes from the query point in MANETs which can reduce the traffic for query processing, as well as maintain high accuracy of the query result. In our methods, queries are transmitted only to neighboring nodes near from the query point to avoid receiving replies from the nodes far from the query point. More specifically, 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 receiving the query replies with the information on itself. Since these methods require neither flooding a kNN query over the entire network nor knowing other nodes' information in advance, they can solve the problems faced by the naïve and existing approaches, which are caused by the bandwidth limitation and dynamical changes of network topology. We conduct simulation experiments to verify that our proposed methods reduce traffic and achieve high accuracy of the query result, in comparison with the existing methods.

In Chapter 3, we propose a method for processing kNN queries that are used to search for the k nearest location-dependent data items in MANETs. In this chapter, we focus on kNN search for data items held by nodes unlike the information on nodes as assumed in Chapter 2. This method achieves low traffic and high accuracy of the query result by limiting the search area. To make the search area small, our method keeps data items at the nodes near the locations with which the items are associated, and nodes cache the data items whose associated locations are near to them. A node issues a query and then the neighboring nodes answer to the query by sending back their copies

without duplicate copies, although each node does not know which data items the other nodes cache. We conduct simulation experiments to verify that our method reduces the traffic involved in processing kNN queries, and also achieves high accuracy of the query result.

In Chapter 4, we propose two convex hull query processing methods; Local Convex Hull (LCH) and Local Wrapping (LW) methods, for reducing the traffic for query processing and maintaining high accuracy of the query result in MANETs. More specifically, in the LCH method, the query-issuing node first floods a convex hull query throughout the entire network. Then, each node replies with information on the nodes that are the vertices of its local convex hul. Here, the local convex hull is the convex hull composed of nodes whose information has already been received. In the LW method, to avoid transmitting queries throughout the entire network, the query-issuing node first sends a convex hull query to a node located on the outer boundary of the network, using a geo-routing technique. Then, the query is basically transmitted only to the nodes on the outer boundary and the query path forms a loop. In this way, unnecessary reply transmissions can be suppressed, even if mobile nodes do not know their neighbors in advance. We also show experimental results for verifying that our proposed methods can reduce the traffic for query processing compared with a naïve method, and also achieve high accuracy of the query result.

Contents

Chapter 1

Introduction

1.1 Background

Recent advances in wireless communication technologies such as IEEE 802.11 [29], Bluetooth [69], Zigbee [89], and Wi-Fi Direct [73] have led to an increasing interest in *mobile ad hoc networks (MANETs)*. MANETs are typical type of networks also known as mobile peer-to-peer networks and composed of only mobile nodes [4, 7, 12, 23, 59, 62]. In MANETs, each node plays a role of router. Even if the source and the destination mobile nodes are not within communication range of each other, data packets are forwarded to the destination mobile node by relaying the transmission through intermediary mobile nodes. Since no special infrastructure is required, many MANET-based applications are expected to be developed in various fields such as military affairs and rescue operations (*i.e.*, collaborating works).

Location-based service (LBS) is typical applications for MANETs, which are typically composed of a large number of nodes over a wide area [9, 20, 77]. In an LBS, real-time location-specific queries for information held by mobile nodes are often used; and in such cases, it is effective to process the queries as k nearest neighbor (kNN) and convex hull queries for supporting collaborating works in MANETs.

KNN queries search for the kNNs (*i.e.*, the k nearest nodes or the k nearest data items associated with a given location (location-dependent data)) from the specified location (*query point*) [5, 15, 16, 18, 21, 32, 61, 63, 76]. By using kNN queries, a node can acquire information near the specified location, which helps to, for example,

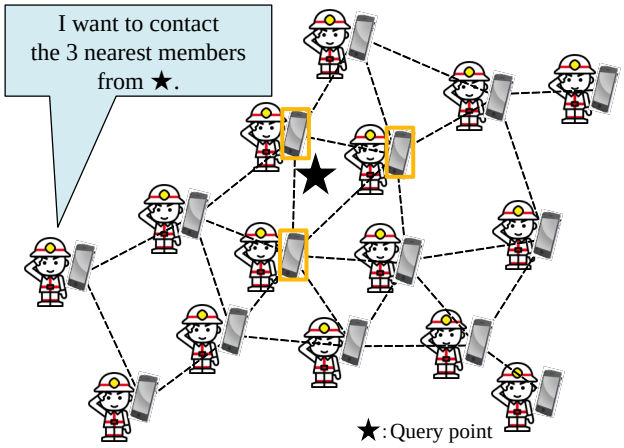


Figure 1.1: Example of a kNN query in a MANET

effectively contact some rescue members near a heavily damaged area in a disaster site in order to request the rescue support. Figure 1.1 shows an example of performing a kNN query, where a rescue worker wants to know the three nearest members from a specific location to contact them.

There is another type of location-specific queries, which is convex hull queries that retrieve the information necessary for calculating the convex hull of all the nodes composing a given network. Using convex hull queries, the query-issuing node can determine the convex hull of the network, which helps figure out the network condition in a real time (*e.g.*, the area in which nodes are present, and the farthest pair of nodes). For example, before a node processes a kind of queries, it determines the parameters for the query processing based on the information acquired from the result of a convex hull query (*e.g.*, the maximum distance between nodes in the network). For another example, in a collaborating work, a team leader can give a caution to a node which moves far away from its team members to avoid the spread of members. In such cases, the node does not need to know the shape of the network in details, and thus, the convex hull query is suitable for achieving it. Figure 1.2 shows an application example of a convex hull query in which a rescue worker attempts to determine the convex hull of nodes (team members) in a disaster site in order to do the rescue work efficiently. To the best of our knowledge, this is the first research studying efficient processing of kNN and convex hull queries in MANETs, which can promote further development of MANET

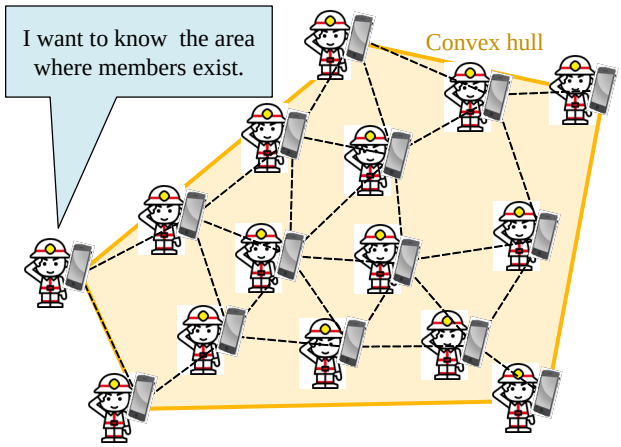


Figure 1.2: Example of a convex hull query in a MANET

applications.

1.2 Research Issues

Location-based services, such as kNN calculation and convex hull detection, have been addressed in many existing works in various types of networks except for MANETs. Most of the works assume that central servers calculate the answers, and focus on designing fast algorithms. As there is no centralized server in MANETs, existing query processing methods based on a centralized server are not applicable to MANETs.

In this section, we describe three research issues for processing location-specific queries including kNN and convex hull queries in MANETs.

1.2.1 Traffic Reduction

In MANETs, since nodes communicate using wireless links where the network bandwidth is limited, the increase of query messages and replies (*i.e.*, traffic) makes packet losses. If packet losses often occur during query processing, the query-issuing node cannot acquire necessary information (*i.e.*, information in an answer set of query processing), which results in decrease of accuracy of the query result.

For example, in a naïve approach, the query-issuing node can calculate the kNNs or convex hull by flooding a query in the entire network [53, 54, 70] and receiving

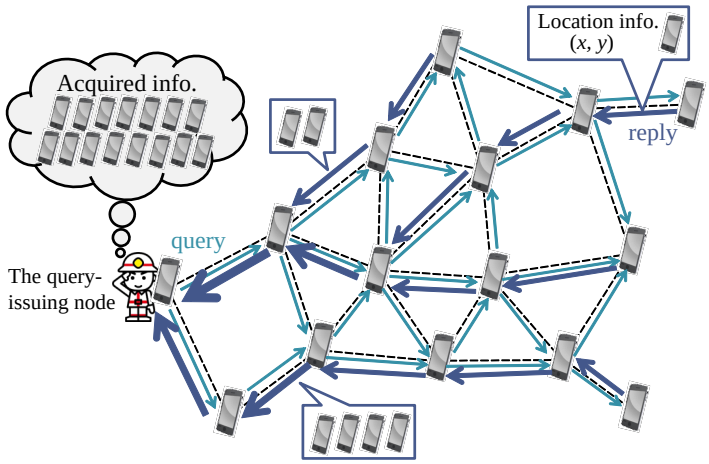


Figure 1.3: Naïve approach for location-based query processing in a MANET

location information on all the nodes. However, many unnecessary replies that are not included in the query result (kNNs or vertices of the convex hull) are sent back to the query-issuing node, and thus, the traffic increases (Fig. 1.3). We should avoid acquiring information on all nodes in the network, since it is a wasteful use of network bandwidth and batteries of the mobile nodes and also may degrade the accuracy of the query result due to packet losses.

Therefore, we need to design query processing methods for reducing traffic while preserving the high accuracy of query results [3, 25, 64, 72]. Our approach is as follows: queries are selectively transmitted only to the nodes that have the information (and data items) included in the query result, and the nodes receiving the query accurately prune unnecessary information based on their own information, the query expression (*e.g.*, the size of the search range), and replies from other nodes.

1.2.2 Query Processing without Using Beacons

Many in-network processing techniques for kNN queries and boundary detection (*e.g.*, convex hull detection) have been proposed for wireless sensor networks (WSNs) [1, 2, 83] where sensor nodes are stationary or location-aware. For example, in [19, 74, 78], the authors proposed kNN query processing methods which are efficient in retrieving kNN information on the fly with low message overhead (traffic) since they do not require the maintenance of indexing structures like R-tree. In their methods, location-

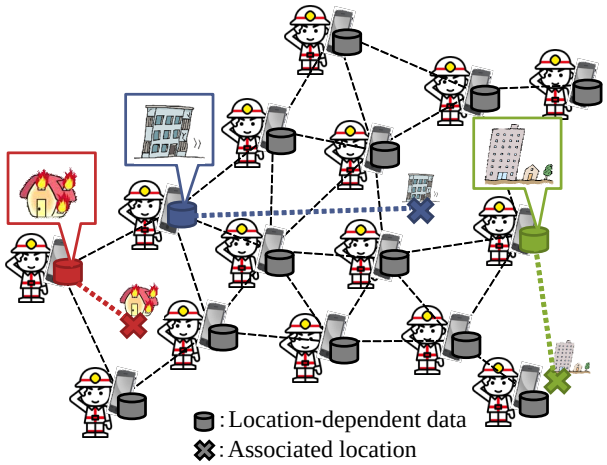


Figure 1.4: Searching for location-dependent data

aware environments are assumed, *e.g.*, mobile nodes must periodically transmit beacon messages (including the location of the sender) to their neighbors to announce changing locations of nodes.

As described above, MANETs possess notable characteristics, such as limitations on network bandwidth, and highly 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 in lower 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, which also decreases the accuracy of the query result.

Therefore, in MANETs it is desirable to retrieve information without using beacons. In this case, since nodes do not acquire the information on neighboring nodes beforehand, effective message exchanges for acquiring neighboring nodes' information are required during query processing.

1.2.3 Searching for Location-Dependent Data Items

In an LBS, it is required to search for not only nodes themselves but also data items associated with a given location (location-dependent data). Fig. 1.4 shows a MANET

where rescue workers hold location-dependent data items (*e.g.*, information about victims and damage of buildings) in a disaster site. As nodes move in MANETs, data items which have been held by a node for a long time are likely no longer related to the node's current location. This is because nodes still hold the data items which are associated with the node's former location (which is generally apart from the node's current location due to node mobility). In such cases, the query-issuing node cannot reduce the area for searching for requested data items by using location information (*i.e.*, methods searching for nodes are not directly applicable). Therefore, effective techniques searching for location-dependent data items are required.

Meanwhile, location-dependent data items can be effectively searched if nodes cache some data items as copies, because the query-issuing node may be able to retrieve requested data items from fewer nodes. Therefore, effective caching of data items can reduce traffic and response time in searching [27, 52, 58, 60, 65].

1.3 Research Contents

In this thesis, we propose kNN and convex hull query processing methods in MANETs which neither acquire all information on nodes within the network nor use beacon, in order to reduce traffic for query processing and maintain the accuracy of query results. The outlines of the proposed methods are as follows.

- Query processing methods for searching k nearest neighbor nodes

  In Chapter 2, we propose two beacon-less kNN query processing methods for reducing traffic and maintaining high accuracy of the query result in MANETs. In our methods, queries are transmitted only to nodes that are near the query point to efficiently acquire kNNs. More specifically, the query-issuing node first forwards a kNN query to the nearest node from the query point by using beacon-less geo-routing method. Then, the nearest node from the query point forwards the query to other close nodes to the query point, and each node receiving the query replies with the information on itself. Our idea for efficient kNN collection is to set waiting times before replying, based on the distance from the respective nodes to the global coordinator. In our 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.

- kNN query processing method for location-dependent data

  In Chapter 3, we address the problem of kNN search for location-dependent data, since search targets for kNN queries are not only nodes themselves as assumed in Chapter 2 but also location-dependent data items such as information about damage of buildings. Here, because nodes holding data items may move far away from locations where the data items are associated, the methods proposed in Chapter 2 are not directly applicable to kNN search for location-dependent data items. Therefore, we propose the method to effectively retrieve the k nearest data items in MANETs. Our method achieves the small search area for query processing by maintaining held data items against nodes' mobility. More specifically, data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose locations are near their own so that the query issuer can retrieve kNNs from nearby nodes, which can achieve low traffic.

- Convex hull query processing methods

  In an LBS, it is also effective to achieve efficient query processing for convex hull queries, which retrieve the information necessary for calculating the convex hull of a given network. In Chapter 4, we address the issue of convex hull query processing in MANETs. As we described above, a naïve approach, where the query-issuing node retrieves location information of all the nodes, produces exceedingly large traffic. Therefore, we propose convex hull query processing methods in MANETs for reducing traffic, while maintaining high accuracy of the query result. In our methods, nodes reply with information on only nodes which are vertices of the convex hull, because the convex hull can be calculated solely on the information on nodes which are vertices of the convex hull of the network. When replying, each node selects the vertex nodes based on the location information of the nodes which have already been replied (without beacons). Since the information on nodes which are not vertices of the convex hull of the network is not included in replies, unnecessary traffic can be reduced than the naïve approach.

1.4 Organization of Thesis

This thesis consists of five chapters, and the remainder of the thesis is organized as follows.

In Chapter 2, we propose two beacon-less kNN query processing methods for reducing traffic and maintaining high accuracy of the query result in MANETs. In our methods, first the query-issuing node transmits a kNN query to the nearest node from the query point (the *global coordinator*). Then, the global coordinator commences the process of acquiring the information on kNNs. In this process, we adopt two different approaches, the Explosion (EXP) and Spiral (SPI) methods to adaptively choose on appropriate method based on some environmental factors such as density of nodes. In the EXP method, the global coordinator floods the kNN query to nodes within a specific circular region, and each node receiving the query replies to the global coordinator with the information on itself. Here, the intermediate nodes can aggregate replies from farther nodes. On the other hand, the SPI method does not require specifying the specific region for sending a query message unlike the EXP method. In the SPI method, the global coordinator forwards the query to other nodes in a spiral manner, and the node that collects a satisfactory kNN result transmits the result to the query-issuing node. We also conduct simulation experiments to verify that our proposed methods reduce traffic and achieve high accuracy of the query result, in comparison with existing methods. Chapter 2 is based on our works published in [40, 41, 46].

In Chapter 3, we propose the Filling Area (FA) method, to process a kNN query which searches for the k nearest location-dependent data items in MANETs. The FA method achieves low traffic and high accuracy of the query result by reducing a search area. To achieve a small search area, the FA method makes data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose associated locations are near them. When a node issues a query, neighboring nodes send back their copies, which will be likely included the query result, with avoidance of duplicate data items being sent back. We also conduct simulation experiments to verify that the FA method reduces the traffic involved in processing kNN queries, and also achieves high accuracy of the query result. Chapter 3 is based on our works published in [42, 43, 44, 47].

In Chapter 4, we propose two beacon-less convex hull query processing methods in

MANETs; the Local Convex Hull (LCH) and Local Wrapping (LW) methods, which perform query processing with low traffic and as well as maintain high accuracy of the query result. In the LCH method, the query-issuing node first floods a convex hull query throughout the entire network. Then, each node replies with information on nodes which are vertices of the local convex hull, which is the convex hull composed of nodes whose information has already been received. In the LW method, to avoid transmitting queries throughout the entire network, the query-issuing node first sends a convex hull query to a node located on the outer boundary of the network, using a geo-routing technique. Then, the query is transmitted only to nodes on the outer boundary, until the query path forms a loop. In this way, unnecessary reply transmissions can be suppressed, even if mobile nodes do not know their neighbors in advance. We also show experimental results verifying that our proposed methods can reduce traffic compared with a naïve method, and also achieve high accuracy of the query result. Chapter 4 is based on our works published in [37, 38, 39].

Finally, in Chapter 5, we summarize this thesis.

Chapter 2

Searching for k Nearest Neighbor Nodes

2.1 Introduction

In LBSs, it is common for nodes to issue kNN queries by which they can acquire the information on the k nearest neighbors (kNNs) from the specified location (query point). Many in-network processing techniques for kNN queries have been proposed for wireless sensor networks (WSNs). The techniques basically assume that nodes exchange beacon messages to accurately know the information on their neighboring nodes. In MANETs, packet losses and packet retransmissions [1] 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 causes unnecessary traffic even when no node is searching kNNs in the network.

Therefore, in MANETs it is desirable to retrieve kNN information without using beacons. In this case, since nodes do not acquire the information on neighboring nodes beforehand, kNN query processing requires on-demand acquisition of neighboring nodes' information. In a naïve approach achieving this, the query-issuing node flooding a kNN query over the entire network and every node receiving the query sends back a reply. However, in this approach, the query-issuing node receives many unnecessary replies not included in the result of the kNN query (kNN result), and thus traffic

[1] In MANETs, messages are generally transmitted with connectionless transmission models.

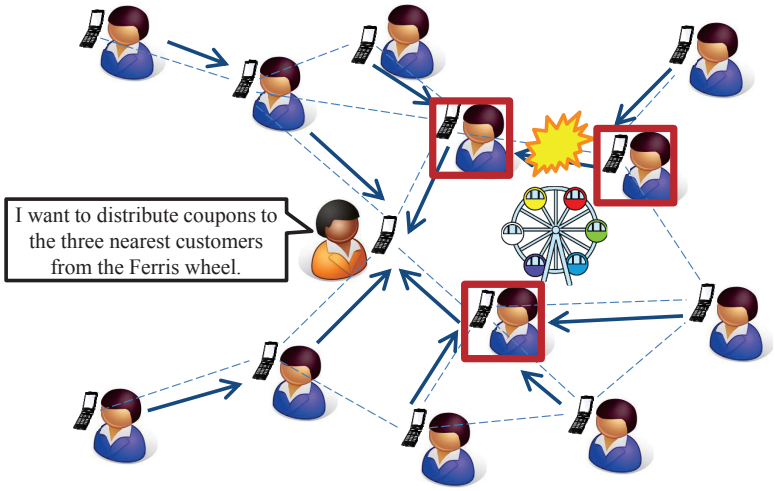


Figure 2.1: Example of performing a kNN query in a MANET

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 in MANETs, the query-issuing node may not be able to acquire all the information on kNNs.

Figure 2.1 shows an example of performing a kNN query, where an event organizer distributes some coupons to kNNs from a specific location. If the query-issuing node receives replies from all nodes as in the naïve method, 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 all the k nearest nodes.

In this chapter, we propose two beacon-less kNN query processing methods in MANETs which aim to reduce traffic during query processing and maintain high accuracy of the query result in MANETs. In these methods, the query-issuing node first forwards a kNN query using geo-routing to the nearest node from the query point (global coordinator). 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 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.

In the EXP method (Figure 2.2), the global coordinator floods the kNN query to the nodes within a specific circular region centered on the query point. This resembles an 'explosion' of the query message from the global coordinator. The size of the circular region is determined based on the density of the nodes in the entire area. If the farther nodes reply earlier with the information on itself to the global coordinator, the intermediate nodes can aggregate replies from farther nodes. Therefore, in the method, the farther nodes from the global coordinator set the lesser waiting time for reply, which is calculated based on the distance between the global coordinator and the node's location (regardless of other nodes' location information). After collecting replies from the nodes in the circular region, the global coordinator replies to the query-issuing node with the kNN result.

The SPI method (Figure 2.3) does not require specifying the specific region for sending a query message unlike the EXP method because it is difficult to accurately determine the region. In the SPI method, the entire area is dynamically partitioned into a set of hexagonal cells. Here, the center of a hexagonal cell is set to the query point and the size of hexagonal cells is determined based on the communication range of the mobile nodes, and thus, each hexagon is autonomously determined. The global coordinator begins forwarding the kNN query to the nearest nodes from the central point of the hexagons which should be visited (*local coordinator*). Each local coordinator collects the information on all the nodes in its hexagonal cell, and then forwards this collected information with the query to the next hexagon in the spiral. The node that surely collects the kNN result transmits the result to the query-issuing node. 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 the node's location. The local coordinator can both acquire the information on nodes within the same hexagon, and determine the next node to which the query should be transmitted. In this method, 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.

The remainder of this chapter is organized as follows. In Section 2.2, we introduce related work. In Section 2.3, we explain our assumption. In Section 2.4, we present our proposed kNN query processing methods. In Section 2.5, we discuss the results of the

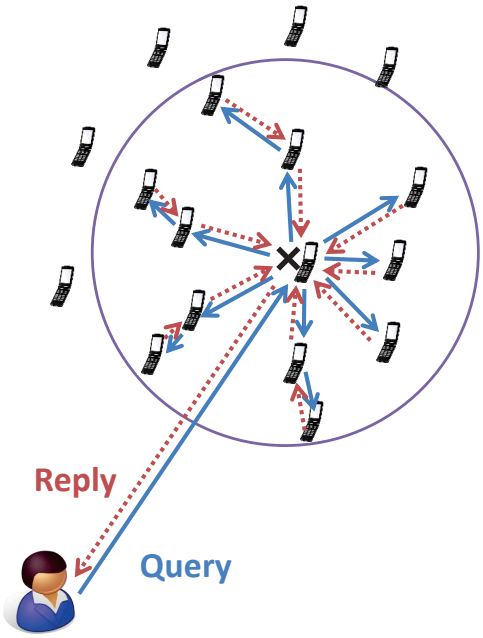


Figure 2.2: EXP method

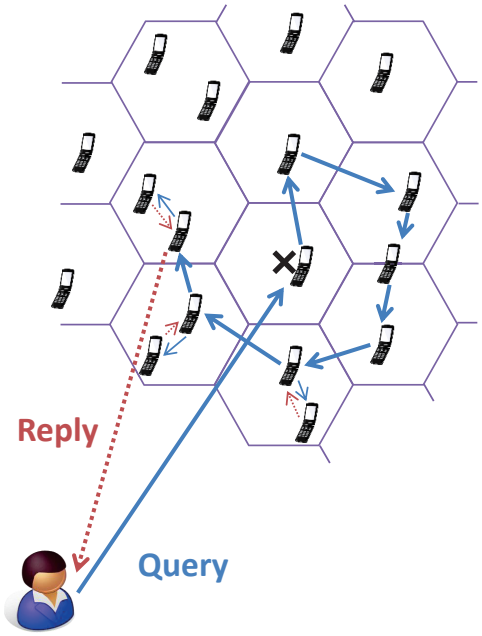


Figure 2.3: SPI method

simulation experiments, and in Section 2.6, we summarize the chapter.

2.2 Related Work

2.2.1 KNN Query Processing in Wireless Sensor Network

Many strategies have been proposed for processing kNN queries in WSNs [31, 50, 81]. Here, we focus on 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.

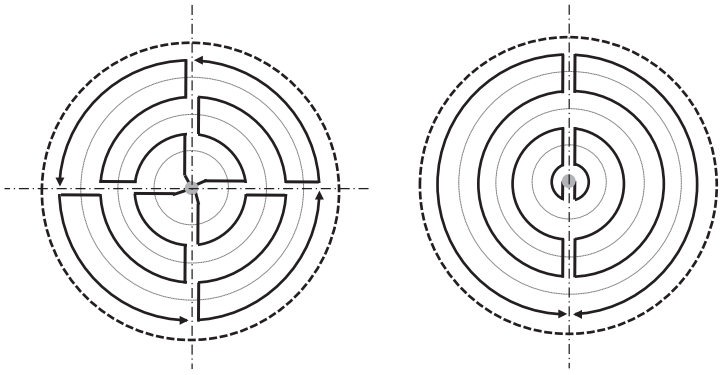
In [19, 74, 78], the authors proposed kNN query processing methods in location-aware sensor networks based on the query propagation method proposed in [79]. In [74], the authors proposed an infrastructure-free kNN query processing method called DIKNN. This method consists of three phases; routing phase, kNN boundary estimation phase, and query dissemination phase. In this method, first a sink node 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 kNN boundary which represents a search range, based on the information on the density of nodes, and partitions the area inside the kNN boundary into sectors. In each sector, a sensor node collects partial results (the information on the nodes within the node's communication range) and propagates the query to the next node in the sector according to a well-devised itinerary structure (Figure 2.4(a)). Finally, the partial results are individually sent back from each sector to the query-issuing node, enabling the query-issuing node to acquire kNNs. In [19], the authors proposed an infrastructure-free kNN query processing method, called PCIKNN, which consists of the same phases as DIKNN's. This method also sets the kNN boundary and partitions it into sectors. With respect to each sector, a sensor node collects partial results along a well-devised itinerary structure, and then, the last node on the itinerary 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 the 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 (*i.e.*, the area inside the kNN boundary) into sectors, the response time of query execution can be reduced even if the search range is large.

In [78], the authors proposed three methods for processing kNN 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 kNN query is propagated along it. The SPI method proposed in this chapter is inspired from the IKNN algorithm that propagates a query along a calculated spiral route (Figure 2.4(b)).

However, these all algorithms 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 kNN query without beacons, a node must process it on the fly using only the node's location information and information in received messages during query processing. In addition, it is necessary to guarantee that the entire range where kNNs are present are fully searched and to identify when to finish the search based solely on the information in a query. Our proposed methods are more suitable to the MANET environments, because they are designed to meet these



(a) DIKNN (b) IKNN

Figure 2.4: Itinerary structures

requirements.

2.2.2 Geo-routing

In various network research fields, many studies on geo-routing have been conducted [34, 48, 51, 75]. Because our proposed methods use geo-routing, we present typical existing geo-routing protocols.

In [35], the authors proposed GPSR which is a geo-routing method in wireless networks. In this method, a node that received a message forwards the message to the nearest node from the query point among its neighboring nodes. In addition, when a node has no closer node from the query point than itself, the query makes detour in order to avoid an obstacle and a network hole. Therefore, this method enables a message to reach the query point even in the case that a relaying node cannot find any nodes closer to the query point than itself. In [49], the authors proposed GOAFR which is a geo-routing method in MANETs. This method is similar to GPSR but is more adaptive to the worst case such as the case that a relaying node has no closer node.

These methods assume the environment where a node knows its neighboring nodes' information by broadcasting beacon messages including its location information, and thus, a node can directly forward (unicast) a message to a specific node using that information. However, in MANETs, since the network topology dynamically changes due to the movement of mobile nodes, each node has to frequently exchange beacon messages,

which causes high traffic. On the other hand, if each node does not frequently exchange beacon messages, a message may not be forwarded to the query point because each node does not accurately know the location of its neighboring nodes (*i.e.*, some of the former neighboring nodes may move out of the node's communication range between beacon intervals). In the MANETs research field, many geo-routing methods without exchanging beacon messages have been proposed.

In [36], the authors proposed a geo-routing method for reducing unnecessary transmission of messages in MANETs. In their method, when a node that received a message is closer than the sender node from the query point, it re-broadcasts the message. Since their method produces multi-paths, the traffic for routing increases. In [28], the authors proposed a geo-routing method for further reducing traffic in MANETs. In their method, the source node broadcasts a message and each node that received the message within a certain forwarding area sets the waiting time based on its location. More specifically, the waiting time is set as a smaller value when the node is closer to the query point. Thus, the node which is closest to the query point re-broadcasts the message and other nodes that overhear the message stop transmitting it. If there is no node in the forwarding area, the message cannot be transmitted to the query point. The method is not suitable for a sparse network. However, this method can suppress producing multi-paths and reduce the traffic in a dense network. Therefore, we adopt the method with some modification for geo-routing in our proposed methods.

2.3 Assumption

The system environment is assumed to be a MANET in which all mobile nodes have the same radio communication function, or capability, whose communication range is a circle of a fixed radius r and messages can be exchanged between any pair of nodes. In the MANET, mobile nodes retrieve the information on other mobile nodes using queries without any network infrastructure. The query-issuing node transmits a query message and acquires the information that should be included in the query result (*i.e.*, kNNs) 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 [2] and i ($1 \leq i \leq n$) is a node identifier. Each mobile node moves freely. Every mobile node knows its current location (x, y) using a positioning system such as GPS, where x and y are the corresponding longitude and latitude, respectively.

2.4 KNN Query Processing Methods

In this section, we present how kNN queries are processed in our methods. In Section 2.4.1, we first present the geo-routing method for transmitting a query to the global coordinator (used in both the EXP and SPI methods). In Sections 2.4.2 and 2.4.3, we explain the details of the EXP and SPI methods, respectively. Finally, in Section 2.4.4, we discuss our methods in terms of the node mobility, nodes' appearance and disappearance, and the setting of the search range for kNNs.

2.4.1 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 [28] to avoid the construction of multiple routes to forward messages. 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 *neighbor search range* which is a circular region centered on the source node (discussed below), it stores the ID of the source node as the parent node and sets the waiting time t for sending a reply, according to the following equation:

$$t = \frac{t_m(\alpha - z)}{\alpha}, \tag{2.1}$$

[2] In this chapter, we assume each node knows n for simplicity. For example, in a rescue operation in a disaster, each member can know the number of members, and in an event site, the event organizer can know the number of customers within the event area. However, in a real environment, it is not always easy to know such information in advance. Therefore, it is useful to remove this assumption (discussed in Section 2.4.4).

where t_m is a positive constant specifying the maximum waiting time before sending a reply, α is the radius of the neighbor search range, and z ($\in R_+$) 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 (see Figure 2.5). Since t is set by the nodes within the neighbor search range (*i.e.*, $\alpha \geq z$), $t \geq 0$. We define the neighbor search range in order to utilize a constructed path to forward a query for the reply. If the neighbor search range is set equal to the communication range, there will be frequent link disconnections between nodes and their parents during query processing, *i.e.*, the chances for utilizing the constructed path decrease. Therefore, the neighbor search range must always be smaller than the communication range. Here, it is difficult (but unnecessary) to appropriately determine the size of the neighbor search range. If the source nodes know the mobility of other nodes, they can construct a stable path for the reply. However, acquiring such information produces large traffic. Moreover, it is not always easy to know (estimate) the node mobility (*i.e.*, the topology during the reply) in advance, since nodes freely change the mobility speed and direction. Therefore, in this method, the radius of the neighbor search range α is determined based on the maximum speed of the source node v_m. The maximum distance that the node moves during a query transmission X is calculated as $X = v_m W$, where W is the search time. Thus, if the radius of the neighbor search range is equal to the communication range minus X, the query path is available unless the nodes move X away from their own parents during a query transmission. Note that our method works well even if α is roughly defined. This is because the node can return the results using geo-routing once again even if a link disconnection between the node and its parent occurs.

According to Equation (2.1), nodes closer to the query point transmit reply messages after shorter waiting times. Thus, the node with the minimum t is the first to transmit a reply message to the source node of the neighbor search message. The nodes that received this reply stop sending a reply if they wait for sending a reply during their own t. The source node of the neighbor search message, having received reply messages from its neighbors, sends a (forwards the) kNN query message only to the node that first sent the reply. The node that receives the forwarded kNN 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 kNNs. In this way, the

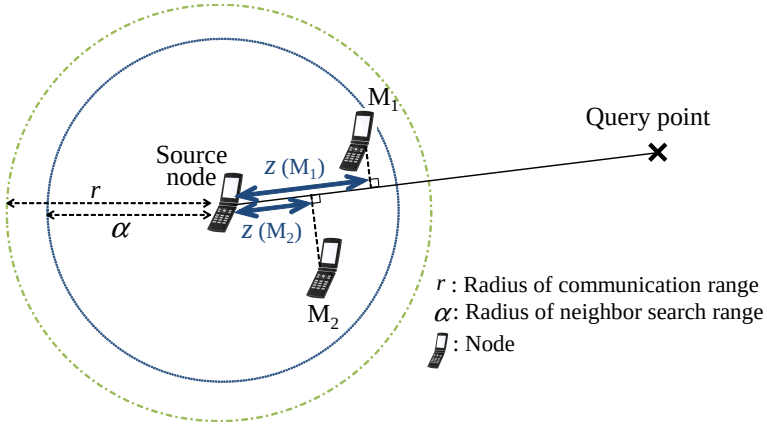


Figure 2.5: Definition of z

query-issuing node can forward a kNN 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 neighbor search range, link disconnections between nodes and their parents rarely occur (only when the nodes move more than $v_m W$ from their own parent during query processing). Thus, this geo-routing method is suitable for query processing in highly dynamic MANETs.

Figure 2.6 shows an example of forwarding a query message using our geo-routing method, when M_1 issues a kNN 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 t. Since M_4 is not within the neighbor search 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 t, is the first to transmit a reply message to M_1, when its t has passed. M_1, on receiving the reply message from M_2, forwards the kNN 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 kNN query.

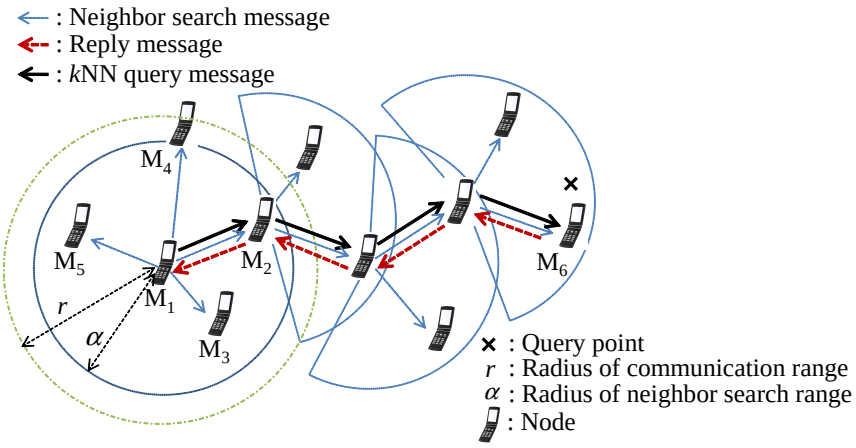


Figure 2.6: Example of executing the geo-routing method

2.4.2 Forwarding a kNN Query and Returning the Result : Explosion (EXP) Method

Concept of EXP method

There must be kNNs 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 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 kNN 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 kNNs) 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 hop-counts, which helps to reduce the traffic. Here, in order for the 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.

We show here the theoretical analysis for verifying cost-efficiency of the EXP method. If packet losses do not occur and the estimated kNN circle is accurately set, *i.e.*, just k nodes exist in the circle, the expected traffic in the EXP method E is calculated as follows:

$$\begin{aligned} E &= (\text{Traffic for query transmission to the global coordinator})H \\ &\quad +(\text{Traffic for acquiring information on } k\text{NNs})k \\ &\quad +(\text{Traffic for reply to the query-issuing node})H, \\ &= (4I+6c)H+(2I+8c+IH')k+(I+2c+kI)H, \\ &= ((2+H+H')I+8c)k+(5I+8c)H, \end{aligned} \tag{2.2}$$

where H is a hopcount from the query-issuing node to the global coordinator, H' is an average hopcount from the nodes within the estimated kNN circle to the global coordinator, I is the byte size of nodes' location information, and c is the byte size of nodes' ID. Note that the upper bound of the traffic in the EXP method equals eE, where e is the number of message retransmission due to packet losses which is a system parameter. As Equation (2.2) shows, the traffic in the EXP method is proportional to k. Here, H and H' are basically calculated based on the communication range and the area size, but the number of nodes in the entire network is not related to E. Therefore, the EXP method can work well regardless of the network density, if the estimated kNN circle is accurately set.

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 kNNs, k, and the query point. Then M_s determines the radius of the estimated kNN circle, R, based on the entire area

size, the total number of nodes in the area, and k, by the following equation:

$$R = \sqrt{\frac{ks}{\pi n}}, \tag{2.3}$$

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

2. M_s transmits a kNN query message to the nearest node from the global coordinator, using the geo-routing method described in Section 2.4.1. 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 kNN is set as k, the radius of the estimated kNN 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 2.4.1, 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 kNNs is set as k, the radius of the estimated kNN 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 kNN circle, it sets the waiting time T for sending a reply, according the following equation:

$$T = \frac{\beta R}{r}\left(1 - \frac{a}{R+b}\right), \tag{2.4}$$

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 (2.4) shows, T decreases as the distance between M_p and M_q increases.

At the same time (without waiting T), 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 kNN circle, it discards the message and does nothing.

5. The node that has set the least T begins to transmit a reply message (after waiting T) 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 kNN result included in the reply message, by adding the information on itself. If the number of nodes whose information is included in the tentative kNN result exceeds k, the information on the node farthest from the query point is removed from the tentative kNN result.

   When T has passed, if the node is not the global coordinator, it transmits a reply message, including the updated tentative kNN 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 T has passed, M_p replies, with the updated tentative kNN result as the final kNN 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 2.4.1, where the query point is set as the location of the query-issuing node.

Figure 2.7 shows an example of executing the EXP method, where M_1 is the global coordinator; and Figure 2.8 shows a flow diagram of the query processing. When M_5 receives a local 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 kNN 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 kNN circle. When T 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 T has passed. Such procedures are performed throughout the entire MANET, until finally M_1 acquires the information on all nodes within the estimated kNN circle. When T has passed at M_1, it transmits the kNN result to the query-issuing node.

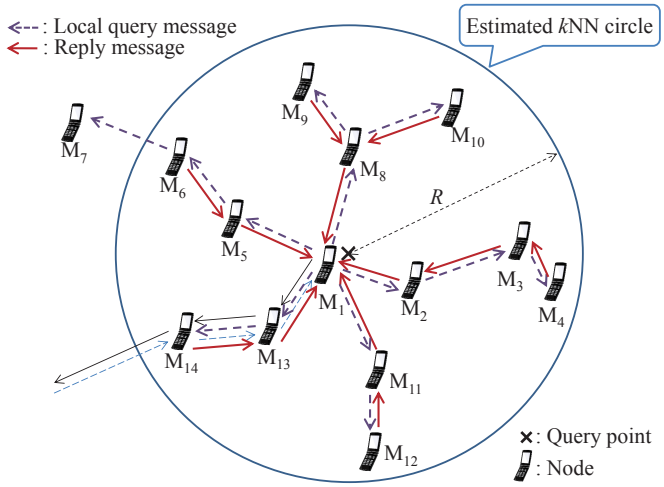


Figure 2.7: Query processing in the EXP method

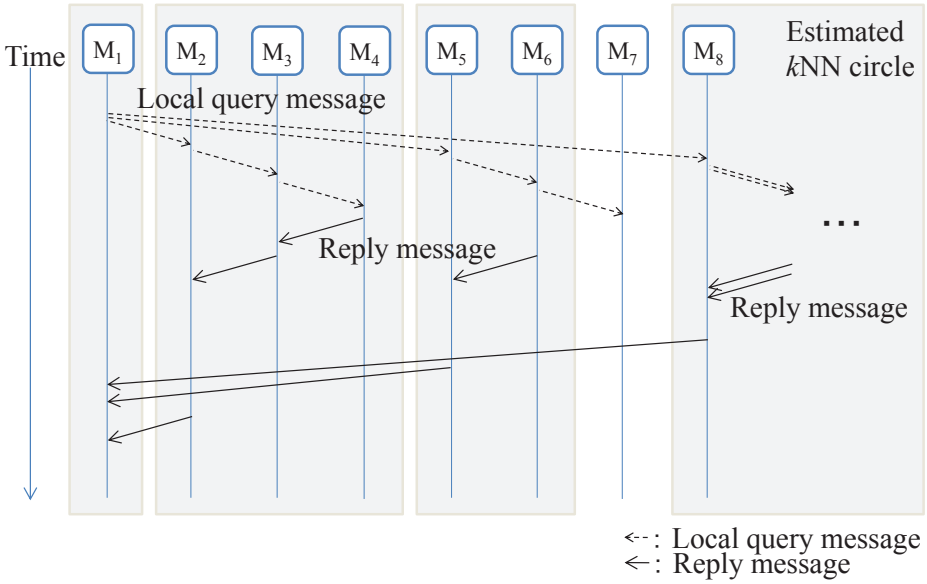


Figure 2.8: Flow diagram of query processing in the EXP method

The EXP method can reduce the traffic required for collecting the kNN 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 kNN circle) is efficiently collected along the tree. However, in this method, the value of radius R must be appro-

priately set. If R is set too low, the information on some kNNs 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.

2.4.3 Forwarding a kNN Query and Returning the Result : Spiral (SPI) Method

Concept of SPI method

As noted in Section 2.4.2, it is difficult to accurately determine the optimal (minimum) radius of the circle which includes kNNs, *i.e.*, excessive traffic and lacks of the information included in the result may occur. Therefore, a method to obtain kNNs without specifying the circle is needed. For this aim, we adopt the strategy of visiting nodes in a spiral manner by following the idea of the IKNN algorithm in [78]. Our strategy aggregates the information of the nodes while a query travels in a single path. It can halt the query process at an appropriate time. The unnecessary information is removed during the query traversal. 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 [19, 74, 78] 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 [19, 74, 78], 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, kNNs 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.

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 kNN 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 (as described in the next paragraph), a query can acquire the information of all 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 2.3, 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.

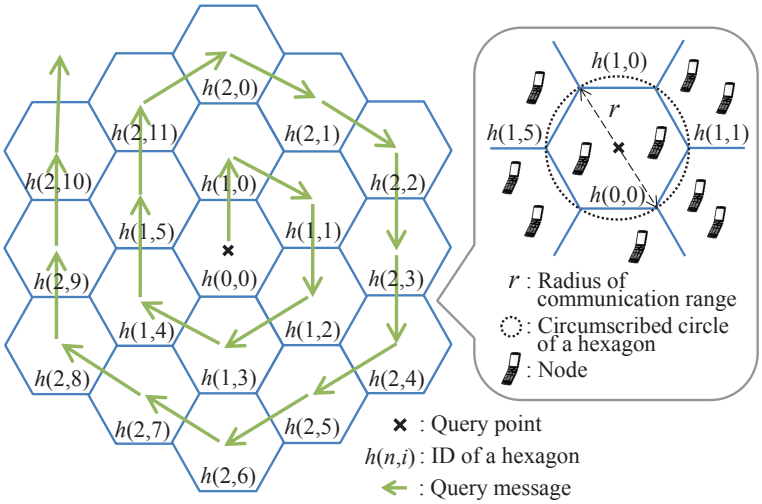


Figure 2.9: Partition of the area

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.

Figure 2.9 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.

Query Processing

In the SPI method, a kNN query is transmitted to a series of hexagonal cells (as described in Section 2.4.3). 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 2.4.1. 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 2.4.1, 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 T for sending a *query request message* (described in Step 6) according to the following equation:

$$T = \frac{t_m h}{r} + \gamma, \tag{2.5}$$

   where t_m 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 (2.5) shows, T decreases with decreasing distance between M_t and the center of the next hexagon.

When T 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.*, $h(n,5n)$ ($n \in \mathbb{N}$), the procedure goes to Step 9.

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 kNNs, k, the query point, the query-issuing node's ID M_s and its location, and the tentative kNN 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 T before sending a query request message (described in Step 6) according to

the following equation:

$$T = \begin{cases} \frac{t_m h_i}{r} & (\text{if (i)}), \\ t_m(1 + \frac{h_{ii}}{r}) & (\text{otherwise}), \end{cases} \tag{2.6}$$

where t_m 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 (2.6) shows, T 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 T 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 kNN result as the final result, to the query-issuing node using the geo-routing method described in Section 2.4.1, 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 kNN result to this node, and the kNN 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 kNN result using the geo-routing method.

Figure 2.10 shows an example of executing the SPI method, where M_1 is the global coordinator, and Figure 2.11 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 kNN result. M_4, on receiving the query message from M_1, transmits a query request message to M_1 when T has passed. When M_1 receives this message from M_4, M_1 transmits a query

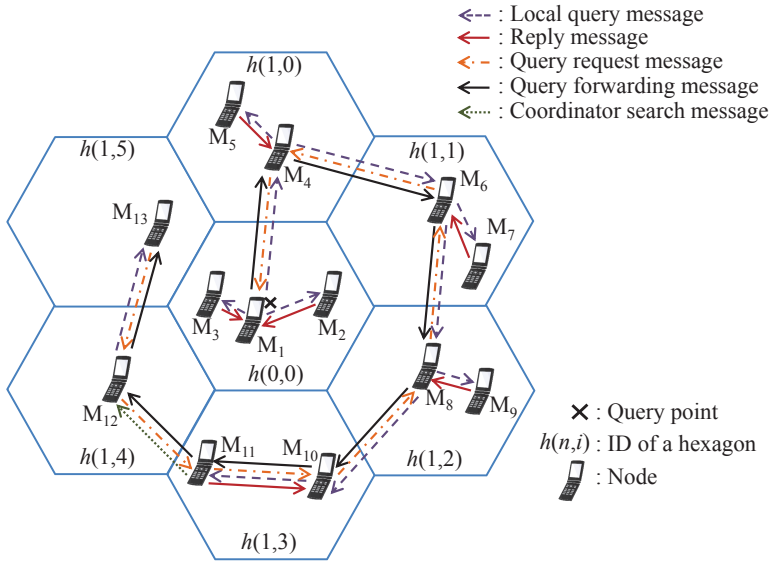


Figure 2.10: Query processing in the SPI method

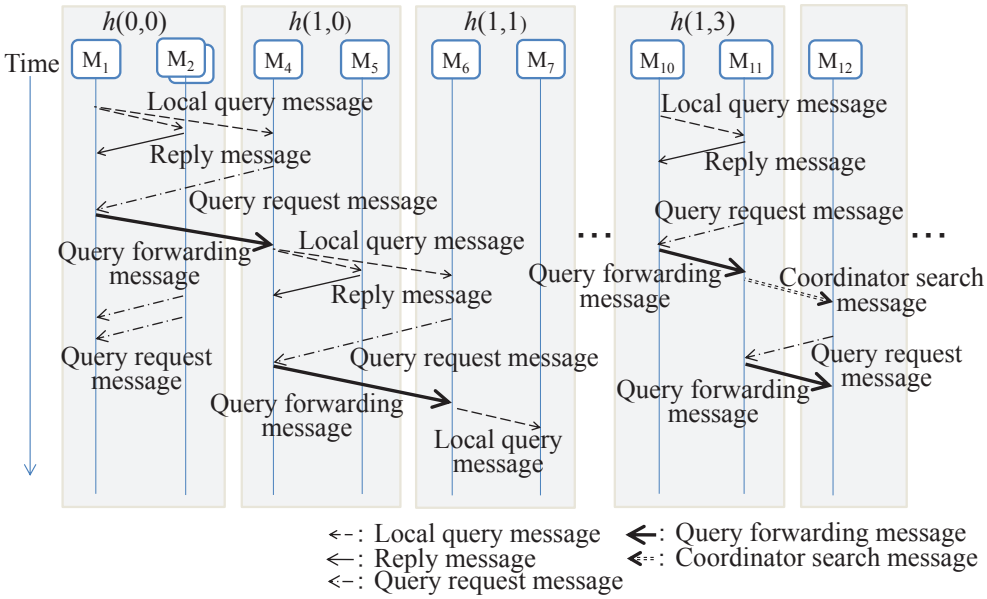


Figure 2.11: Flow diagram of query processing in the SPI method

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 kNNs and initiates a reply to the query-issuing node.

The SPI method expands the searching area until a node surely collects the information on kNNs. 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 kNNs in most cases. However, due to message collisions, the query-issuing node may not always find kNNs.

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 $2.5t_m$ 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 kNNs. Therefore, it may be difficult to maintain high accuracy of the query result [3].

2.4.4 Discussion

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

[3]Note that 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.

handshakes without exchanging beacon messages, nodes select the next hop using the neighbor search 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 kNNs) to the query-issuing node by using geo-routing once again. In our simulation experiments, the graphs regarding node speed (Figure 2.14(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).

Appearance and Disappearance of Nodes

In this chapter, 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).

Estimated kNN Circle in the EXP method

The EXP method determines the search range (estimated kNN 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 kNN circle is not very accurate, and thus, its performance may degrade.

Specifically, if the estimated kNN 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 kNNs 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.For example, in [45], we proposed the EXP based methods for kNN query processing in MANETs where the density of nodes is ununi-

form; the One-Hop (OH) and Query Log (QL) methods. These methods can set the size of the estimated kNN circle more appropriately using the information acquired during the query execution even if each node cannot know the information on the area size and the total number of nodes beforehand, and the density of nodes in the entire network is not uniform. In the OH method, the nearest node from the query point acquires its neighbors' location and then determines the size of the estimated kNN circle. On the other hand, in the QL method, a node which relays a reply of a kNN query stores the information on the query result for future queries. During query forwarding, the query-issuing and query-relaying nodes attach some of the stored information to the query, which is used to estimate the density of nodes near the query point.

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 chapter, 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 kNN 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 kNN 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 kNNs dynamically change). That is why we aim to balance between accuracy and traffic, rather than achieving perfect accuracy with too much (unacceptable) overhead.

2.5 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[66].

2.5.1 Simulation Model

The number of mobile nodes in the entire system is n (M_1, $\cdots$, M_n). These mobile nodes are present in an area of 1,000[m] $\times$ 1,000[m], and move according to the random waypoint model [8], 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]. Messages are transmitted via UDP. Packet losses and delays occur due to radio interference. We assume that each node knows its current location. The query point specified by a kNN query is (500, 500). The t_m in Equations (2.1) and (2.5) is set at 0.1 based on our preliminary experiments, so that our methods achieve good performance. α in Equation (2.1), β in Equation (2.4), and γ in Equation (2.5) are respectively set to 97, n/500, and 0.005n/30, based on our preliminary experiments.

We compare the performance of our proposed EXP and SPI methods with that of a naïve method, DIKNN[74], and the EXP and SPI methods using beacons. In the naïve method, which does not use beacons, the query-issuing node broadcasts a kNN query to its neighbor nodes, and then the query is transmitted to nodes within a specific region using the geo-routing method proposed in [36]. Specifically, each mobile node that receives the kNN 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 kNN circle (which is the same as that in the EXP method). In addition, if a node that receives the kNN query is within the estimated kNN 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 [74], 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 [4]. In Section 2.5.4, we evaluate two different cases, with beacon periods of 10 and 20[sec] respectively.

The requested number of kNNs, 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 2.5.2 and 250 in Section 2.5.3. In addition, to evaluate the impact of node mobility, v is varied from 0.5 to 20[m/s] in Section 2.5.4, 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 kNN query. We repeat this process 1000 times (*i.e.*, 1000 queries) for every 20 seconds, and evaluate the following three criteria.

- Traffic

  We examine the total volume of query messages and replies exchanged in processing a query. Table 2.1 shows the size of messages used, in our methods and the comparative methods, where f denotes the number of nodes whose information is included in the reply, and p 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 computation and message transmissions. In our approaches, message transmissions predominate in the search for kNNs,

[4] 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 2.1: Message Types and Sizes

Type	Size [B]
Query (naïve method)	48
Reply (naïve method)	24
Neighbor search (geo-routing)	48
Reply (geo-routing)	8
Query (EXP method)	56
Query (SPI method)	48
Query (DIKNN method)	56+24p
Local query (EXP, SPI and DIKNN methods)	56
Coordinator search (SPI method)	24
Query request (SPI method)	8
Query forwarding (SPI method)	72+16f
Reply (EXP, SPI and DIKNN methods)	32+16f
Ack to the received reply	8
Beacon	32

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 kNN result. In the naïve 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 kNNs whose information is included in the kNN result acquired by the query-issuing node, to the requested number of kNNs, k. We define "accuracy of query result" as the MAP (Mean Average Precision) value, which measures the performance of the result with a ranking [55]. The

MAP is an average of the AP (Average Precision) of the respective queries. The AP and MAP are determined by the following equations:

$$AP_i = \frac{1}{k} \sum_{j=1}^{k} \frac{g}{j} e, \tag{2.7}$$

$$MAP = \frac{1}{q} \sum_{i=1}^{q} AP_i, \tag{2.8}$$

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, q is the total number of issued queries (*i.e.*, 1,000 in this simulation), and e is determined by the following equation:

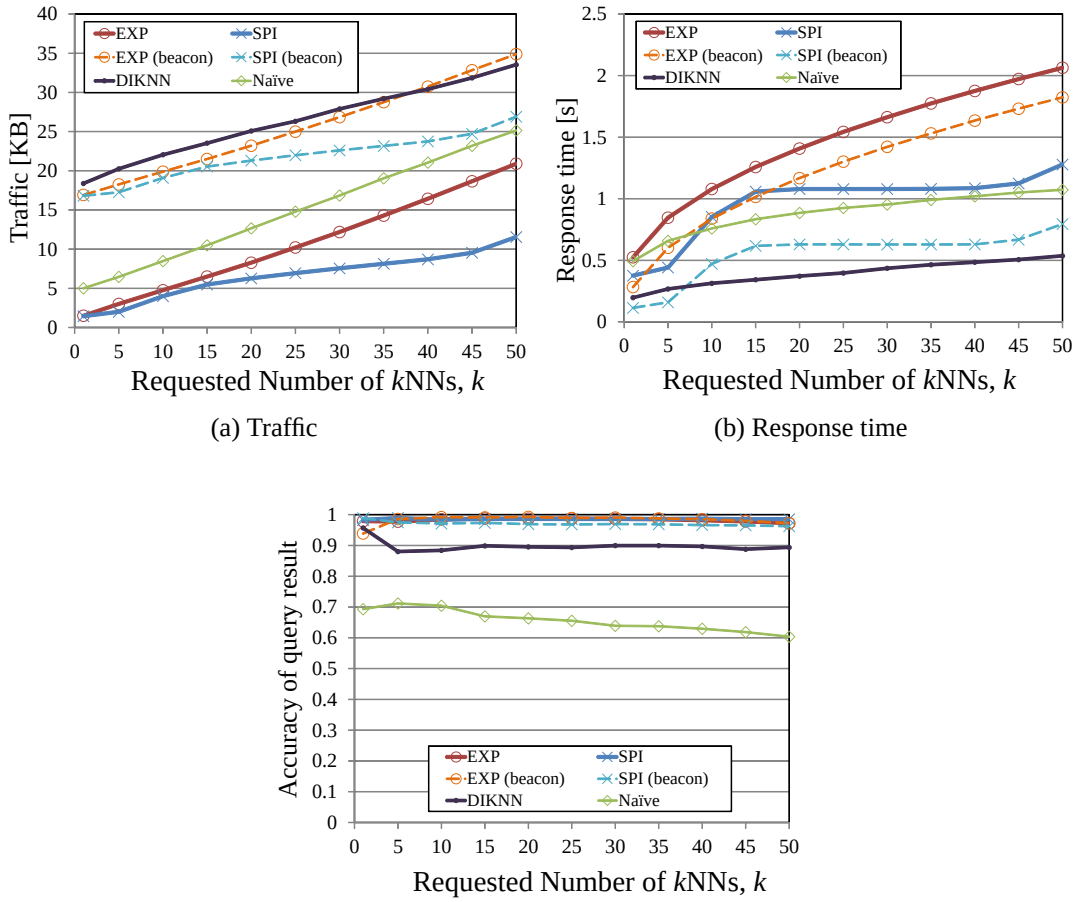
$$e = \begin{cases} 1 & \text{(if the } j\text{-th nearest node is included} \\ & \qquad \text{in the } k\text{NN result).} \\ 0 & \text{(otherwise).} \end{cases} \tag{2.9}$$

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

2.5.2 Simulation Results in the Case of 500 Nodes

First, we examine the performance of our proposed methods when the number of nodes is 500. Figures 2.12 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 Figure 2.12(a), the response time in Figure 2.12(b), and the accuracy of query result in Figure 2.12(c).

From Figure 2.12(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 naïve method, because in the former, a kNN



(a) Traffic

(b) Response time

(c) Accuracy of query result

Figure 2.12: Impact of k in the case of 500 nodes (high node density)

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 Figure 2.12(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.*, t), and thus the response time is increased. In the EXP method in particular, every node must wait for T before sending back a reply, which results in an increase in response time. In the naïve 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 naïve method is roughly similar to that of the SPI method.

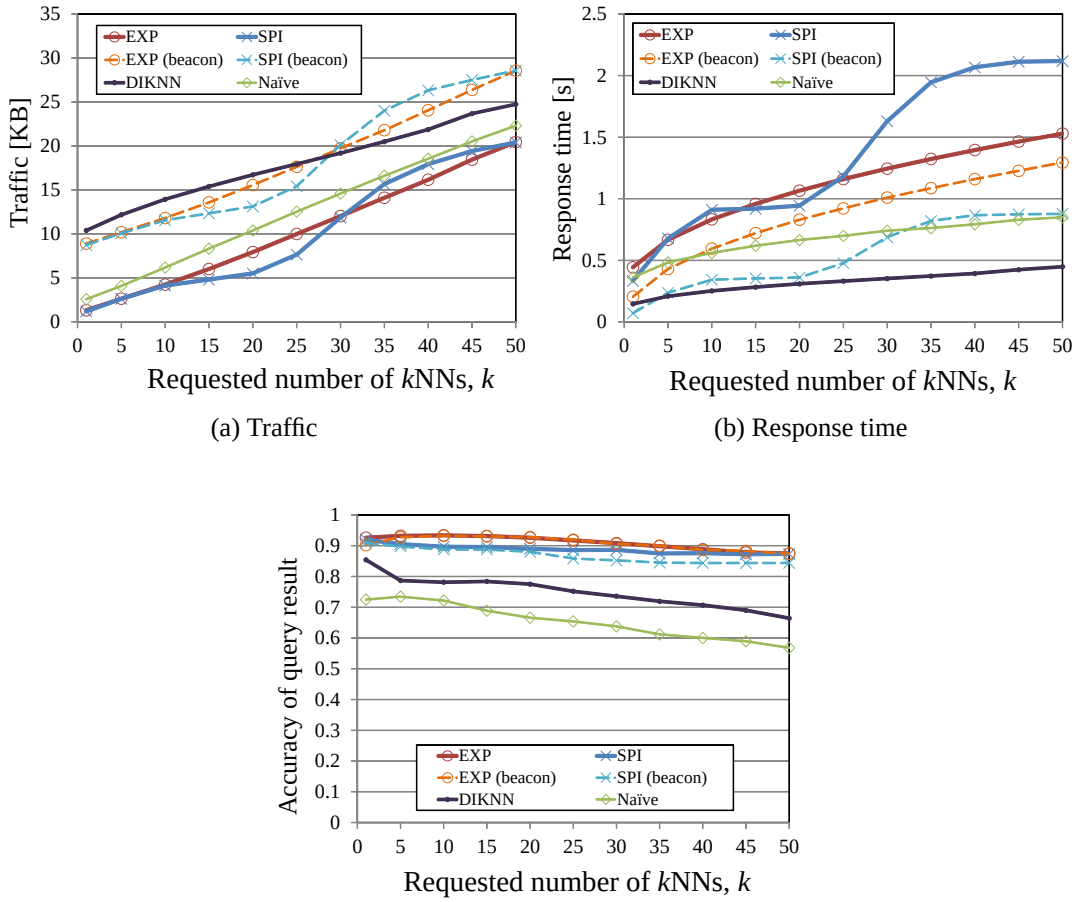
From Figure 2.12(c), the accuracy of query result is very high (nearly 1) in our methods, in contrast to the lower accuracy of query result in the DIKNN and naïve methods. This is because, in the latter, packet losses often occur due to individual replies, from all sectors in the DIKNN method, and from all nodes in the naïve method.

2.5.3 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). Figure 2.13 shows the simulation results. In these graphs, the horizontal axis indicates the requested number of kNNs, k, and the vertical axis indicates the traffic in Figure 2.13(a), the response time in Figure 2.13(b), and the accuracy of query result in Figure 2.13(c).

From Figure 2.13(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 naïve 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.

From Figure 2.13(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,



(a) Traffic

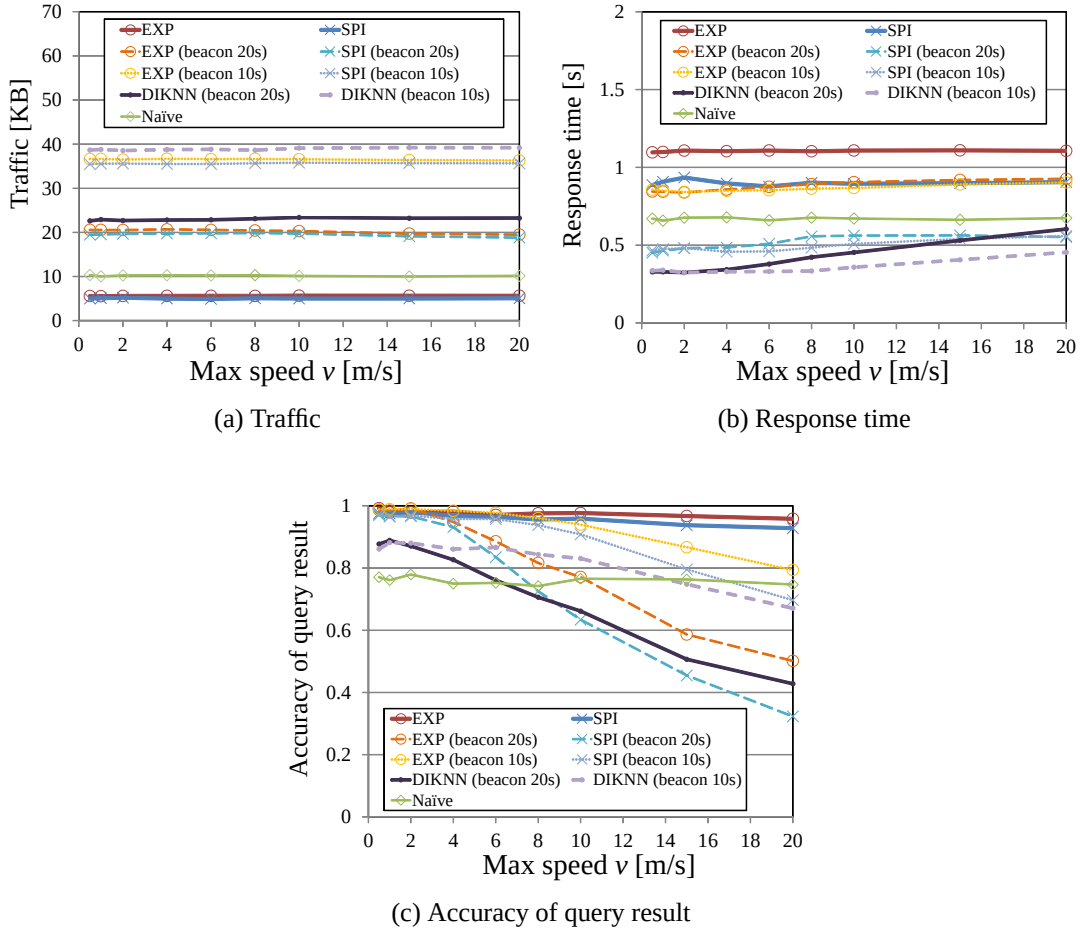
(b) Response time

(c) Accuracy of query result

Figure 2.13: Impact of k in the case of 250 nodes (low node density)

the response time is less than in the case of 500 nodes, because some of the parameters determining waiting time (*e.g.*, β) were based on node density (lessened for low density) in order to avoid message collision.

From Figure 2.13(c), none of the 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 that with 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 fewer than in the other methods.



(a) Traffic

(b) Response time

(c) Accuracy of query result

Figure 2.14: Impact of node mobility

2.5.4 Impact of Node Mobility

Finally, we examine the performance of our proposed methods by varying node speed when the number of nodes is 500. Figures 2.14 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 Figure 2.14(a), the response time in Figure 2.14(b), and the accuracy of query result in Figure 2.14(c); "(beacon xs)" denotes a method using beacons with a period of x seconds. In these simulation experiments, k is set at 10.

From Figures 2.14(a) and 2.14(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 Figure 2.14(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 Figure 2.14(a). Thus, we can observe that the methods using beacons are not suitable for a highly dynamic network.

2.5.5 Discussion

Comparison among Six Methods

The respective methods have advantages in different applications or network conditions. Table 2.2 compares the features of the respective methods.

In the methods employing beacons ("yes" at Beacon column in Table 2.2), the frequency of the broadcasting period of beacon messages directly affects the performance of the methods. If nodes frequently exchange beacon messages, kNNs can be searched with high accuracy of the query result, because nodes can accurately know the information on their neighbors. However, it involves much traffic, which often makes the accuracy of the query result of the query result worse than beacon-less methods. In terms of 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 2.2) 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 2.2). 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.

Table 2.2: 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
Naïve	no	(Broadcast)	-	Proactive

How to Choose One from the Two Proposed Methods

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

In this chapter, 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 naïve 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. With such estimation, the query-issuing node may be able to select an appropriate method from either EXP or SPI method.

2.6 Conclusions

In this chapter, we have proposed kNN query processing methods in MANETs which aim to reduce traffic for query processing and maintain high accuracy of the query result. 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 the naïve 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.

Chapter 3

Searching for k Nearest Location-Dependent Data Items

3.1 Introduction

In an LBS, it is effective to search for the k nearest data items associated with a given location (location-dependent data) from the query point. Figure 3.1 shows an example of performing a kNN query in a MANET, where a rescue worker acquires the k nearest data items (*e.g.*, information about victims and damage of buildings) from his/her current location in a disaster site. If the query-issuing node receives all the data items within the entire network, a substantial amount of unnecessary traffic is generated.

Meanwhile, location-dependent data items can be effectively searched if nodes cache some near-by data items, because the query-issuing node may be able to retrieve kNNs from nodes within a small region. Therefore, effective caching of data items can reduce traffic and response time in searching. Here, in kNN search, the query-issuing node often specifies its own location as the query point (*i.e.*, the query-issuing node retrieves nearby information). In this case, it is efficient for nodes to cache data items whose locations are near their own.

As nodes move in MANETs, data items which have been cached by a node since long time ago are likely no more members of kNNs from the node's current location, because the cached data items are associated with the node's former location. Hence, in order to make good use of cached data items for a kNN query, it is necessary to maintain

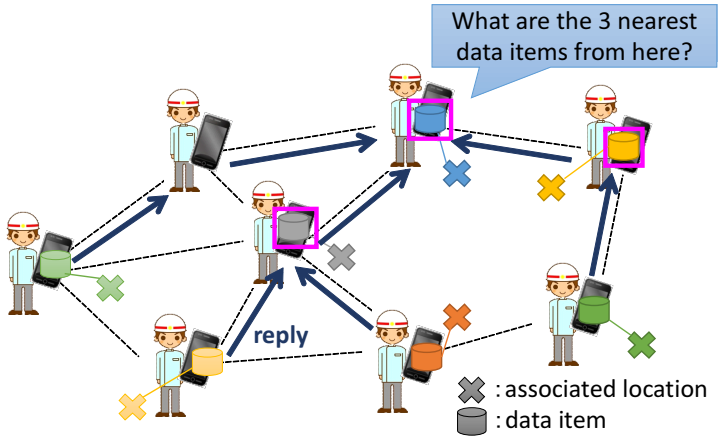


Figure 3.1: Example of a kNN query for location-dependent data items in a MANET

cached data items to adapt to location changes. In addition, when the requested number of data items k is greater than the number of cached data items, the query-issuing node needs to acquire data items included in the kNNs from neighbors by query processing. However, if the neighbors reply with their own cached data items, much unnecessary traffic will occur because such neighbors often cache the same data items and thus reply with duplicate data items. Furthermore, without centralized management of data items (*i.e.*, when there is no server), it is difficult for query-issuing nodes to know whether acquired data items are kNNs from the query point (*i.e.*, the correct query result) unless nodes know all data items. Therefore, it is important to achieve a mechanism for a query-issuing node to know that it actually has acquired kNNs.

In [57], the authors proposed a kNN query processing method for a pure mobile P2P environment (*i.e.*, MANET). In their setting, nodes collaborate in answering the query, and acquire kNNs with less communication cost than that of centralized methods. Their work addresses a similar problem to that of our study described in this chapter, but the work in [57] assumes that each peer always cache correct kNNs from its current location, and does not focus on the problem of replies with duplicate data items.

In this chapter, we propose the Filling Area (FA) method to process a kNN query which searches for the k nearest location-dependent data items in MANETs. The FA method achieves low traffic and high accuracy of the query result. To the best of our knowledge, this is the first kNN query processing method maintaining cached data items

against nodes' mobility and preventing replies with duplicate data items in a dynamic distributed system including a MANET. In the FA method, when a node issues a kNN query, it acquires data items retained by nodes within a specific region (*search area*), which is the circular area centered at the query-issuing node's location.[1] When nodes within the search area reply with data items (original and cached) which are likely included in the query result, they avoid duplicate replies with the same data items by overhearing messages sent by the other nodes.

Nodes reply with data items (original and cached) which are likely included in the query result, avoiding replies with duplicate data items by overhearing messages. To minimize the search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items (as copies) whose locations are near their current locations. In particular, when a node retaining data items moves beyond a given *data boundary*, away from the location with which the items are associated, it forwards the items to the nodes nearest to the location with which they are associated. We also summarize experimental results which verify that our method can reduce traffic in comparison with existing methods and achieve high accuracy of the query result.

The remainder of this chapter is organized as follows. In Section 3.2, we introduce related works. In Section 3.3, we explain assumptions of this work. We present our kNN query processing method in Section 3.4. In Section 3.5, we show the results of the simulation experiments. Finally, we summarize this chapter in Section 3.6.

3.2 Related Work

As described in Section 3.1, there have been no studies that directly address the problems targeted in this chapter. In this section, we introduce some existing studies on location-dependent data management in LBS and kNN query processing, which are somewhat related to our work.

[1] In this chapter, the query point is the current location of the query-issuing node. We assume that the query-issuing node retrieves nearby information.

3.2.1 Location-Dependent Data Management in LBS

In [26], the authors proposed a line-based data dissemination protocol for sensor networks. Sinks disseminate data items to nodes within a vertical virtual line which divides the field into two parts, and nodes search for data items there. This method does not assume kNN query processing and replication of location-dependent data. In [71], the authors proposed the Skip-Copy method, specifically aimed at managing location-dependent data items in MANETs. This method sparsely distributes copies of location-dependent data items to increase data availability in a wide range. However, in this method, nodes access a single location-dependent data item, in contrast to our method's aim of acquiring multiple location-dependent data items near a specific point. In [82], the authors proposed a system to provide an LBS which does not depend on pre-established infrastructure. The system uses a mobile agent that remains within a certain geographical area, moving among mobile nodes in the area. Our method is inspired by this approach in keeping data items within a specific region.

3.2.2 kNN Query Processing

In [68], the authors proposed a method for efficiently acquiring kNNs from mobile query points. This method can reduce disk access search costs for databases, when the query point is moving and kNN results change. It assumes that the information of all static objects (such as hospitals or schools) has been previously obtained (*i.e.*, a centralized method). In [10], the authors defined Continuous All k-Nearest Neighbor (CAkNN) queries which continuously identify all nodes' closest neighboring nodes, and proposed a 'Proximity' algorithm for efficiently processing such queries in smartphone networks. This algorithm only works well in areas covered by a set of network connectivity points (*e.g.*, cellular towers for cellular networks). Moreover, each node must regularly report its positional information to the query processor, which is too costly in MANETs. In [80], the authors proposed a method to continuously monitor kNNs in a wireless sensor network. Here, sensors detect objects moving around the target region, and these sensors collaborate to continuously monitor the kNNs (k nearest objects) from the query point. However, since this method assumes that the sensors are statically deployed, it is not applicable to MANETs.

In Chapter 2 [40], we have proposed a kNN query processing method for MANET. In this method, a node floods a query to nodes within a specific circular region centered on the query point, and each node receiving the query replies with information on itself. Thus, this method can avoid flooding in the entire network. However, the search object in this method is the k nearest nodes (not location-dependent data items). Searching location-dependent data is more challenging because data held by nodes may move far away from their associated locations.

In [87], the authors proposed a collaborative caching framework for spatial queries in mobile P2P networks. The study presented the Signature Augment Tree (SAT), which is based on the R-tree, to maintain the information on data items cached by neighbors. Therefore, the query-issuing node can acquire needed data items by sending queries only to nodes caching the data items, which reduces the data traffic involved in the acquisition of data items. However, this framework basically assumes that nodes request data items to the server when they cannot acquire the data items locally. In addition, nodes must have complete information on data items cached by neighbors, whereas our method can work without such information.

3.3 Assumption

We basically assume the environment described in Section 2.3. In addition, we assign a unique *data identifier* to each data item in the system. The set of all data items in the system is denoted by $D = \{ D_1, D_2, \cdots, D_m \}$, where m is the total number of data items[2], and D_j ($1 \leq j \leq m$) is a data identifier. Each data item includes *data point*, which is the location information associated with the data-generating location. For simplicity, all data items are assumed to be the same size. Nodes have a cache storage capacity of C data items.

3.4 KNN Query Processing Method

In this section, first we describe the design policy of our proposed method. Then, we describe how sensor data items are managed in the FA method. Finally, we describe in

[2] m may dynamically change due to appearance, update, and disappearance of data items.

detail how kNN queries are processed in the FA method.

3.4.1 Design Policy

Since there is no centralized server in a MANET, a centralized approach is inappropriate. For example, if data items were transmitted to a specific node as soon as they were generated, the query-issuing node could acquire kNNs by transmitting a query to the specific node because the specific node would have all the data items and know the kNNs for all queries (*i.e.*, all query points). However, this is not realistic owing to the nodes' limited storage capacity, *i.e.*, the size of all the data items in the network is too large for each mobile node to store. Moreover, the transmission of numerous queries to the specific node would cause a temporary spike in traffic volume in the vicinity of the node. Therefore, we have designed a distributed kNN processing method.

In an LBS, users often require location-dependent data items from near their own location; and when using a kNN query, they repeatedly require the kNNs from their own location. Therefore, a method is needed in which nodes retain data items whose location is close to their own, keeping in mind that nodes' locations will change as they move. Note that kNNs from the own location change along node mobility.

3.4.2 Maintenance of Data Items

Original Data Item

When a node retaining an original data item moves beyond a given *data boundary*, d, away from the *data point* with which the item is associated, the original item is transmitted to the node nearest its data point, using our geo-routing method (an extension of the protocol proposed in [28]), which adopts a three-way handshake protocol to send a data item to the neighboring node nearest the data point. By repeating this procedure, the data is forwarded to the node nearest its dependent location.

More specifically, in our geo-routing method, the node which retains a data item first broadcasts a *neighbor search message* when it moves beyond d. Then, when a node receives the neighbor search message, if it is closer to the data point than the source node, it sets a waiting time for sending a reply. Because nodes closer to the data point transmit a reply message after a shorter waiting time, the nearest neighboring node from

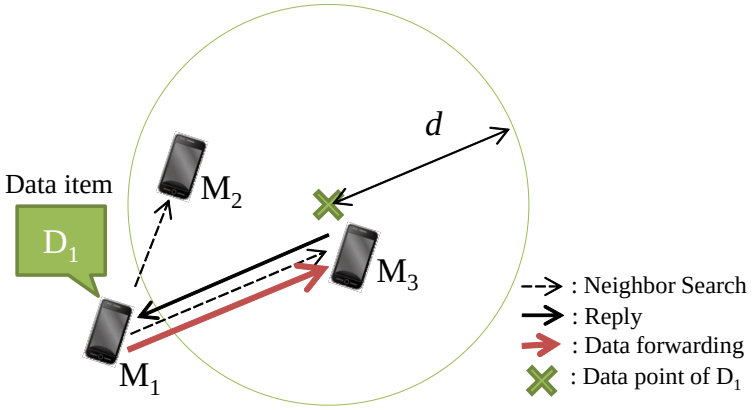


Figure 3.2: Example of forwarding an original data item

the data point is the first to transmit a reply message to the source node. Other nodes which have overheard the reply message, do not reply to suppress unnecessary traffic. The source node that has received reply messages from its neighbors sends a (forwards the) *data forwarding message*, including the data item, only to the node that first sent the reply. The node that receives the data forwarding message broadcasts a neighbor search message using the same procedure. Finally, if a node that has sent a neighbor search message receives no reply messages when the data point is within its communication range, it recognizes itself as the nearest node from the data point (Fig. 3.2).

Through the above procedure, a node which retains a data item can forward the item to the nearest node from the data point with low traffic because this geo-routing method requires neither the use of beacon messages nor the construction of multiple paths. This achieves that a node can acquire a data item by searching nodes from which the distance to the data point of the item is less than d.

Figure 3.2 shows an example of executing data transmission in which M_1 retains a data item D_1. When M_1 moves farther than d from the data point of D_1, it broadcasts a neighbor search message. When M_2 and M_3 receive the neighbor search message from M_1, they set a waiting time for sending a reply because they are closer to the data point of D_1 than M_1. M_3 sets a shorter waiting time because the waiting time is based on the distance from the data point of D_1 to a given node's location, and M_3 is closest to the data point of D_1 among the nodes receiving the message. When its waiting time has passed, M_3 initiates a reply to M_1. On receiving the reply message from M_3,

M_1 forwards a data forwarding message, including D_1, to M_3. Then, M_3 transmits a neighbor search message in the same manner as M_1. Finally, M_3, on receiving no reply, retains D_1 until it moves farther than d from the data point of D_1.

Cached Data Item

Requirements: As described above, since users repeatedly require the kNNs from their own location using kNN queries, it is useful for nodes to retain the CNNs from their locations. In such a situation, if k is less than C, a node can know the kNNs among its cached data items without query processing. Even if k is greater than C, a node often can acquire kNNs by seeking data items from only nearby nodes, because these nodes retain nearby data items.

In MANETs, it is a considerable challenge to maintain current CNNs, because the location of a given node (in this case the query point) changes due to node mobility, resulting in changes in the CNNs. If nodes frequently exchange messages including data items, they can update the CNNs from their new location, but it significantly consumes the network bandwidth, causing frequent packet losses and high energy consumption. On the other hand, when nodes do not frequently exchange messages, their own cached data items become much different from actual CNNs, resulting less useful for kNN query processing.

Another feature is that perimeter nodes cache almost same data items since each node caches data items whose locations are near their own. If all nodes in the network maintain their own CNNs, the traffic for maintaining CNNs within the network is large due to maintenance of almost same data items on respective neighboring nodes. Therefore, in such a case (*e.g.*, the density of nodes is high), it is effective that selective nodes rather than all of the nodes maintain cached data items.

Cache Maintenance Procedure: In the FA method, to maintain current CNNs, a node updates its retained CNNs by exchanging messages with its neighboring nodes. Here, we define the circular area in which there are CNNs, as the node's *assignment*, whose center point is the location at which the node updates its CNNs (*update point*), and whose radius is the distance from this update point to the farthest data point among cached data items. For maintaining cached data items, nodes newly cache its current CNNs when a node moves farther than d from its update point, unless neighboring

nodes already cache data items whose data points are close from the node's current location.

First, when a node generates a data item, it signals the occurrence of the item by flooding a message including the item, over a specific area, which should be appropriately determined such that the data item is surely sent to nodes which retain it as a CNN, but not sent to an unnecessary wide area. After this, the nodes that received the message retain the data item in their caches if necessary. Algorithm 1 shows the procedures involved in maintenance of cached data items.

When a node moves farther than d from its update point, it broadcasts a request message to its neighboring nodes, which includes the IDs of data items cached by the node, and the current location of the node. Each node receiving the request message sets a waiting time w for a response based on the distance d' from its update point to the issuer of the message, using the following equation: $w = \alpha\ d'$, where α is a parameter for waiting responses. The waiting time decreases, as the update point of a node nears the location of the message issuer. This is effective because a nearer node likely caches more data items which the issuer will cache as CNNs. The node which has set the least waiting time broadcasts a *Maintenance-Stop message* if d is less than μ and the distance between its current location and its update point is less than λ. Otherwise, it selects the candidates, which are data items closer than the C-th nearest data items from the location of the request issuer, and at the expiration of its waiting time it broadcasts a response message including the candidates. If a node receives a Maintenance-Stop message, it cancels sending response. The request issuer does not update cached data items, and it transfers to the Maintenance-Stop (MS) mode and sets the MS time S (the left case in Fig. 3.3). On the other hand, nodes overhear response messages from each other (the right case in Fig. 3.3), and if the node has no candidates which are not sent by other nodes, it does nothing, resulting in a reduction of unnecessary traffic.

Data updating

In the real environment, data items are constantly generated, updated, and deleted, and it is important for LBSs to easily manage such data updates. KNN queries are sensitive to data updating because the kNN answers to queries may change. Data updating patterns are divided into those that are location dependent and those that are not. An example of

Algorithm 1:
Maintenance of cached data items

```
1: // node moves farther than d from its update point or S has passed at node
2: if node is in MS mode then
3:     recover from MS mode
4: end if
5: send Request

6: // node receives Request
7: if node is not in MS mode then
8:     store ID of request issuer's cached data items
9:     set waiting time
10: end if

11: // waiting time has passed at node
12: if node has not received Maintenance-Stop then
13:     if there exists near request issuer then
14:         send Maintenance-Stop
15:     else if there are candidates then
16:         send Response(candidates)
17:     end if
18: end if

19: // node receives Response
20: if node is not in MS mode then
21:     if node is request issuer then
22:         update cached data items
23:     else
24:         store ID of candidates
25:     end if
26: end if

27: // node receives Maintenance-Stop
28: if node is request issuer then
29:     set S
30: end if
```

the former may be seen in fixed-point sensor nodes periodically observing updates to data items, including present value. In this case, although the kNNs do not change, because the data points remain the same, nodes must replace cached data items with new data items whose data points show the same location. On the other hand, when data items are generated independently of location (*e.g.*, when mobile nodes periodically sense data items at a given location), the kNNs may change.

A naïve method to notify other nodes of data updates is to flood messages, including new data items, over the entire network, which causes excessive traffic. Our data maintenance method can easily handle data updating, whether location dependent or not; for here, both original and cached data items are retained by nodes near the respective data points. This means that when new data items are generated, nodes generating such items can send the notification message including the items over only the limited area near themselves. Therefore, our data maintenance method is suitable for location-dependent data items with data updation.

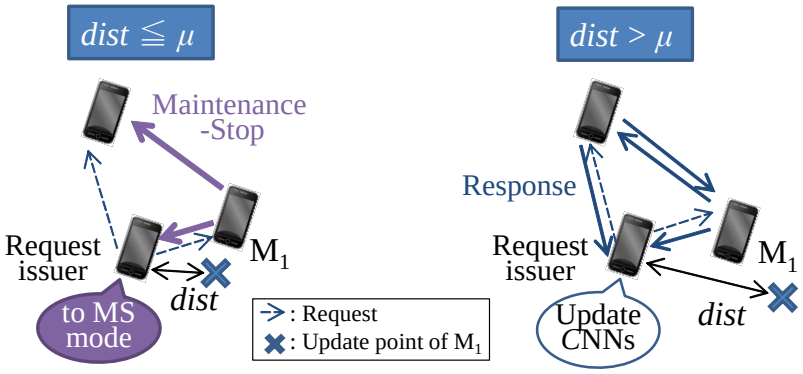


Figure 3.3: Example of cache maintenance

3.4.3 Forwarding kNN Query and Replying with Result

In the FA method, when a node issues a kNN query, it floods the query to nodes within a specific region (*search area*), which is a circular area whose center point is the query point. Nodes, which is not in the MS mode[3], reply with data items which will be likely included in the kNNs, avoiding replies of the duplicate data item by overhearing messages from other nodes. Algorithm 2 shows the procedures involved in query processing. In this pseudo code, *Assignment (node)* denotes the node's assignment.

First, the query-issuing node specifies the requested number of kNNs, k, and determines whether its cached data items satisfy the kNNs from the query point (line 2 in Algorithm 2). More specifically, if $k \leq C$, and the data points of the k nearest data items among data items cached by the query-issuing node are present within the *guaranteed area* which is a circular area centered on the query point, whose radius is $l - U$ where l is the radius of Assignment (query-issuing node) and U is the distance between the query point and its update point, the query-issuing node can complete the search. This is because the kNNs are guaranteed to be included in the guaranteed area; that is, the data point of any data item outside this circular area must be farther from the query point than the data points of data items within this area (see Fig. 3.4). Otherwise, the query-issuing node performs query processing for acquiring kNNs in cooperation with nearby nodes.

The query-issuing node determines the radius of the search area, R, based on the

[3]In the FA method, nodes in the MS mode do nothing in query processing, which reduces traffic. Note that even if a node in the MS mode, it can issue a query, and procedures of the method is same.

Algorithm 2: Query processing

```
1:  // query-issuing node begins to search
2:  if Assignment (query-issuing node) guarantees kNNs then
3:      complete search
4:  else
5:      calculate search area
6:      send Query
7:      set T'
8:  end if

9:  // node receives Query for the first time
10: if node is within search area and not in MS mode then
11:     store ID of source node as parent
12:     store Assignment (query-issuing node)
13:     send Query
14:     set T
15: end if

16: // node receives Reply or EmptyReply issued by node N
17: if node is query-issuing node then
18:     store data items
19: else if node is not in MS mode then
20:     store Assignment (N)
21:     update T
22:     if node is parent of source node then
23:         send Reply or EmptyReply to parent
24:     end if
25: end if

26: // T has passed at Node
27: if Assignment (Node) is not covered then
28:     if no data within Assignment (Node) then
29:         send EmptyReply to parent
30:     else
31:         send Reply to parent
32:     end if
33: end if

34: // T' has passed at query-issuing node
35: if k data items are acquired and area of kNNs is covered then
36:     complete search
37: else
38:     calculate new search area
39:     send Query
40:     set T'
41: end if
```

information on data items cached by itself, and k, using the following equation:

$$R = l(\beta\sqrt{\frac{k}{C}} - 1) + d, \tag{3.1}$$

where β (≥ 1) is a parameter to avoid missing some kNNs due to misestimation of the density of data items[4]. According to Equation (3.1), R is principally determined based on the density of data items cached by the query-issuing node (*i.e.*, $C/\pi l^2 = k/\pi L^2$ [$1/m^2$], where L is the radius of the *Filling Area* which is the circular area in which data points of the kNNs are present with high probability). Because each node caches

[4]When $k \geq C$, R surely becomes positive. Otherwise, in an unlikely event that the calculated R has a negative value, β is set a greater value to make R positive.

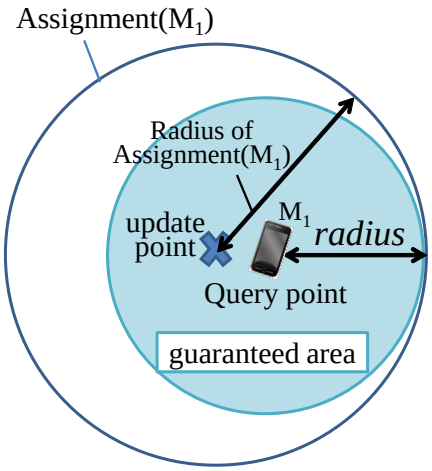


Figure 3.4: Guaranteed area

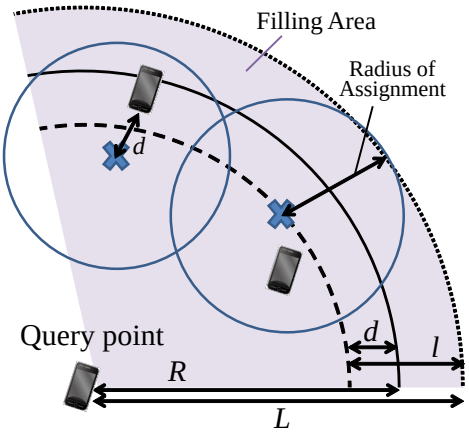


Figure 3.5: Definition of R

data items, it can reply with the data items whose data points are roughly l away from its location, and thus nodes within the circular area of the inner dotted line in Fig. 3.5 will have data items within the Filling Area. Therefore, R can be reduced by l from L. Moreover, since each node locates (at most) d away from its update point, R is enlarged by d.

Then, the query-issuing node floods the search area with a kNN query message which includes the query-issuing node's ID; the remaining requested number of kNNs, k' (*i.e.*, $k -$ the number of data items within the guaranteed circular area); R; the query-issuing node's assignment (*i.e.*, its update point and l); and the query point (*i.e.*, its own location). At the same time, the query-issuing node sets the waiting time T' for awaiting replies, calculated by $\delta\lceil R/l \rceil$, where δ is a positive constant (line 7 in Algorithm 2).

If a node receiving the query message (the receiver) is within the search area and not in the MS mode, it sets the reply delay T (the time from receiving a query for the first time to sending a reply), using the following equation:

$$T = \frac{\delta P(A_m - A)}{A_m}. \tag{3.2}$$

$$P = \begin{cases} 1 & (\text{if } u < l). \\ \lfloor u/l \rfloor & (\text{otherwise}). \end{cases} \tag{3.3}$$

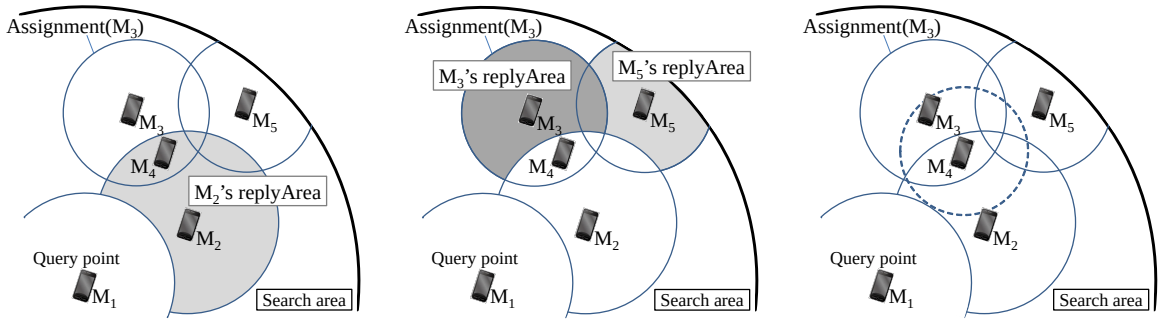
$$
\begin{aligned}
A \;=\; & Assignment\ (receiver) \\
& \cap\ \overline{Assignment\ (query{-}issuing\ node)} \\
& \cap\ \overline{Assignment\ (\forall N)}.
\end{aligned} \tag{3.4}
$$

Here, N is a node that has already issued a reply received by the receiver, A_m is a system parameter which is a positive constant, and u is the distance from the query point to the receiver's update point. In Equation (3.2), T decreases as u decreases and A increases[5]. From Equation (3.4), A indicates the size of the area where data items which have not been replied yet, called *replyArea* (*e.g.*, the gray-colored area in Fig. 3.6). As A increases, the priority of replying gets higher because more data items are present in the area with high probability, and thus the node can reply with these items. Nodes within a smaller P reply earlier every additional l, because nodes with update points nearer to the query point can reply with a smaller hopcount to the query-issuing node.

A node within the search area initiates a reply message (after waiting T), including candidate data items and its assignment, to the query-issuing node if A is non-zero; that is, if Assignment (this node) is not fully covered by replies from other nodes (Line 27 in Algorithm 2). The candidate data items are the (at most) k' data items which have not yet been sent in reply by other nodes, among the data items whose data points lie within the Filling Area. If the node has no candidates, it transmits an *empty reply message* to the query-issuing node, instead of a reply message. The empty reply message includes only its assignment (not data items). With the reception of the empty reply message, the query-issuing node can recognize that there are no data items in the assignment attached to a message, and thus, the method guarantees that the kNN query exhaustively searches the search area. When a node receives (or overhears) reply or empty reply messages, it recognizes the area covered by the replying node, avoiding multiple replies of the same data item; and if the overhearing and replying nodes' assignments overlap, A is updated, resulting in an updated T.

The query-issuing node receiving a reply message updates the tentative kNN result. When T' has passed, if the number of acquired data items exceeds k and the circular area

[5]In an unlikely event that the calculated T has a negative value, though A_m is inherently greater than A, T is set at 0.



(a) M_2's turn (b) M_3's and M_5's turns (c) M_4's turn

Figure 3.6: Example of FA method

centered on the query point, whose radius is the k-nearest data point, is fully covered with assignments of all replying nodes, the search is completed and the query-issuing node updates its cached data items with the query result. If not, the query-issuing node re-estimates R (*e.g.*, β is decreased) and repeats the same process, to avoid decreasing the accuracy of the query result; note further that there is an upper limit on the number of such repetitions (not endless).

Figure 3.6 shows an example of executing query processing in which M_1 issues a kNN query, and nodes' update points correspond to their respective locations for simplicity. All nodes receiving the kNN query (M_2-M_5) set respective Ts because they are within the search area. First, M_2 initiates a reply to M_1 because its assignment is larger and the distance between the query point and itself is less than that of the other nodes (Fig. 3.6(a)). Nodes overhearing M_2's reply message (M_3-M_5) store M_2's assignment, and update their respective Ts in light of this. After that, M_3, and then M_5, transmit a reply (or empty reply) message (Fig. 3.6(b)). M_4 does nothing (when its own T has passed) because its assignment is fully covered by assignments of other nodes (Fig. 3.6(c)).

With this method, unnecessary transmissions of queries and replies can be suppressed, as nodes search within a limited search area and overhear the replies of other nodes.

3.4.4 Discussion

In this chapter, we assume that the query point is the location of the query-issuing node. However, this is not always the case in a real environment, as it sometimes happens that a user wishes to acquire kNNs from a specific location far from the location of the query-issuing node. The FA method can be easily adapted to such a situation as described below. If the query-issuing node specifies, as the query point, a location different from its current location, first, the query is transmitted to the nearest node from the query point, using geo-routing [28, 36]. Then, the nearest node from the query point executes the FA method as the method's query-issuing node, and replies with acquired kNNs to the query-issuing node (through the query path from the query point or using geo-routing).

3.5 Simulation Experiments

In this section, we show the results of simulation experiments evaluating the performance of our proposed method. For the simulation experiments, we used the network simulator QualNet5.2 [66].

3.5.1 Simulation Model

The number of mobile nodes in the entire system is 500. These mobile nodes are present in an area of 1,000[m] $\times$ 1,000[m], and move according to the random waypoint model [8], with a movement speed and pause time of from 0.1 to v[m/s] and 200[s], respectively. 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 above setting on the network is basically same as that in Section 2.5.1.

The number of data items in the entire system is 250, and the size of data items is 1024[B]. Specifically, randomly selected 250 nodes respectively generate data items associated with their own location once per 300[s], and each data item is valid for 300[s] since it is generated, *i.e.*, the number of data items in the entire network is constantly

250 even though data items are dynamically generated and disappeared. Nodes have a cache storage capacity of C data items. The query point specified by a kNN query is the location of the query-issuing node. d, α, MS, λ, μ, and β in Equation (3.1), and δ in Equation (3.2) are respectively set to 30, 1, 30/v, 10, 30, 1.1, and 3 based on our preliminary experiments.

We compare the performance of the FA method with that of six alternate methods in which only original data items (not copies) are maintained in the same manner as in the FA method, *i.e.*, the original data items remain at nodes near their respective data points. The first method is the EXP method for searching location-dependent data items (extending the EXP method in [46]). We assume that each node knows the total number of data items, and the entire area size. Thus, in the EXP method, the radius of the search area, R, is determined by the following equation:

$$R = \sqrt{\frac{ks}{\pi m}} + d, \tag{3.5}$$

where s is the area size (s = 1,000×1,000), and m is the total number of data items (m = 500). The query-issuing node floods a query to nodes within the search area, and nodes receiving the query reply with data items whose data points are within the circular area centered on the query point, whose radius is $R-d$. "EXP" in the graphs denotes this method.

The other five methods, based on the FA method for detailed comparison, differ only in terms of the maintenance of cached data items. In these methods, query processing is performed in the same way as in the FA method, with the exception of the cached data maintenance described in Section 3.4.2. In the first method (denoted by "FA(all)" in the graphs), all nodes respectively maintain CNNs (*i.e.*, without the SM mode), which is proposed in [44]. Therefore, in this method, all nodes maintain CNNs even when the density of nodes is high (*i.e.*, perimeter nodes respectively cache almost the same data items). In the second method (denoted by "fixed CNN" in the graphs), nodes statically retain CNNs from the initial update point (*i.e.*, the update point is fixed). This method does not require the cached data maintenance, but due to mobility, nodes tend to retain data items in their cache whose data points are far from their current locations, and thus, assignments of neighboring nodes are little overlapped with each other. Therefore, in this method, the mechanism for avoiding multiple replies containing the same data items (*i.e.*, overhearing and canceling replies) in the FA method may not work well. In

addition, when new data items are generated, the new data items are transmitted over the entire area, since nodes generating these items cannot know where nodes whose assignment contains the location of the new items are present. The comparison with this method demonstrates the effect of updating cached data items as CNNs from nodes' current locations.

The third method (denoted by "casual cache" in the graphs) maintains cached data items in an ad hoc manner. In this method, a node updates cached data items when it incidentally receives or overhears a message including data items nearer to its current location than its own cached data items. Then, it updates the update point as its current position and regards its own C data items as its CNNs. Thus, this method does not ensure that cached data items are actual CNNs from the node's update point. In other words, it may often happen that the node misses some of CNNs, which causes the node misjudging the density of data items lower and its assignment larger than the real ones. In query processing, as the same in the FA method, query-issuing nodes determine the search area based on their own cached data items, resulting in misjudgement of geographical density of data items.

The last two FA-based comparison methods involve periodical maintenance of cached data items. In these methods, data items are periodically sent to nodes to notify them of the presence of the data items, and the receiving nodes update their cached data items. To examine the impact of the maintenance interval, we set the interval at 5[s] and 300[s], respectively (denoted by "periodic (high)" and "periodic (low)", respectively, in the graphs).

In evaluating all the methods, the query processing procedures for each query are performed only once (*i.e.*, with no repetitions of a search), in order to evaluate the performance of the methods at with one iteration. Table 3.1 shows the parameters and their values used in the simulation experiments. The parameters are set by default at the constant values to the left of their respective parenthetical range, and varied over this range in the simulations.

In the FA based methods, first, nodes flood data items so that others can retain them in the cache storage. After three hundred seconds have passed since the simulation started, the query-issuing node, randomly chosen among all nodes, issues a kNN query. We repeat this process 500 times (500 queries) for every 0.5 seconds, and evaluate the three criteria which is the same described in Section 2.5.1 (*i.e.*, traffic, response time,

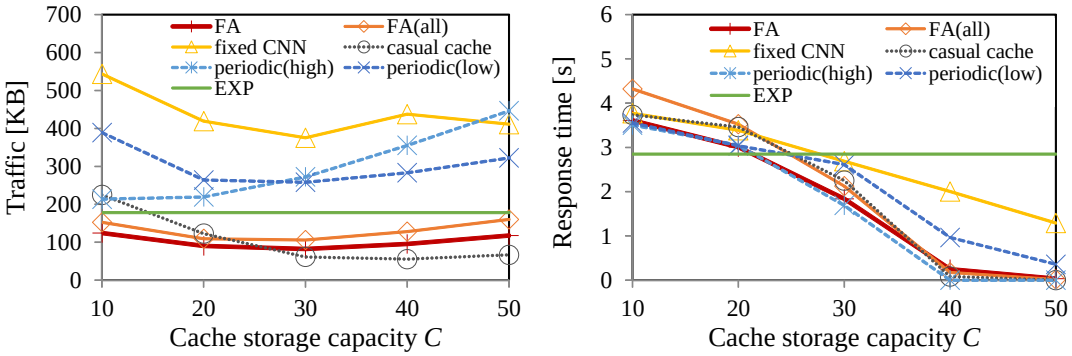
Table 3.1: Parameter configuration

Symbol	Meaning	Values (Range)
C	Cache storage capacity	20 (10 - 50)
k	Requested number of data items	30 (1 - 50)
v	Max speed	1.0 [m/s] (0.5 - 2)

Table 3.2: Message types and sizes

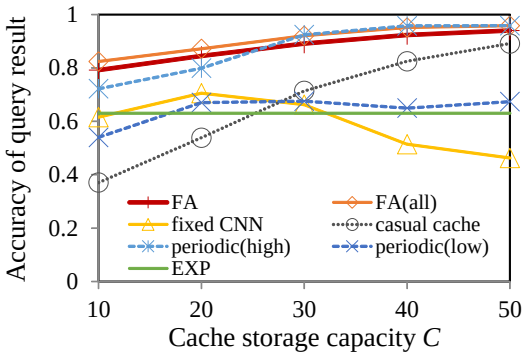
Type	Size [B]
Neighbor search (data)	48
Reply (data)	16
Data forwarding (data)	40+1024
Notification (data)	32+1024
Request (data)	32
Response (data)	24+1024t
Maintenance-Stop (data)	16
Query (FAbased)	104
Query (EXP)	64
Reply (FAbased)	64+1024q
Reply (EXP)	32+1024q
EmptyReply (FAbased)	56
Ack for a reply (all)	16

and accuracy of the query result). The traffic in this section includes the messages for query processing, that for data flooding when data generation, that for maintenance of original data, and that for cached data maintenance. Table 3.2 shows the size of messages in the FA method and the comparative methods, where q and t respectively denote the number of data items included in the reply, and that included in the response for the request. Here, “Neighbor search (data)” denotes a neighbor search message in the maintenance process of data items described in section 3.4.2, and so on.



(a) Traffic

(b) Response time



(c) Accuracy of query result

Figure 3.7: Impact of the number of cached data items C

3.5.2 Impact of Cache Storage Capacity

In the FA method, the cache storage capacity affects performance. Therefore, we first show the results of simulations in which the capacity of cache storage, C, is varied. Figure 3.7 shows the simulation results. In the graphs, the horizontal axis indicates the cache storage capacity, and the vertical axes respectively indicate the traffic in Fig. 3.7(a), the response time in Fig. 3.7(b) and the accuracy of the query result in Fig. 3.7(c).

From Fig. 3.7(a), in the FA method and the FA method by all nodes, when C is less than 30, as C increase, the traffic decreases, because the number of data items sent in reply during query processing decreases (*i.e.*, the query-issuing node has more kNNs in

its cache). On the other hand, when C is more than 30, as C increase, the traffic increase since the traffic involved in data maintenance increases (*i.e.*, the number of data items exchanged in data maintenance increases). In the FA method, the traffic is smaller than in the FA method by all nodes because the traffic involved in data maintenance is small since some nodes are in the MS mode, which stop operations for the maintenance of cached data items during a MS period. In the FA method with fixed CNNs, the traffic is greater than that in the FA method because duplicate data items are often sent in reply from multiple nodes. This is because cached data items are less relevant to their associated locations due to nodes' mobility, *i.e.*, though nearby nodes tend to cache dissimilar data items, remote nodes often cache the same data items. In such a case, even if the assignments of those remote nodes are overlapped, they cannot overhear messages sent by the others, and thus, the mechanism for avoiding multiple replies containing the same data items (*i.e.*, overhearing and canceling replies) does not work well, resulting in large traffic. Moreover, in this method, the traffic involved in notification of data generation is very large because messages including new data items are transmitted over the entire area. This result shows the effectiveness of our data maintenance mechanism, in which nodes near the data points of data items retain those items.

In the FA method with casual caching, the traffic is basically small because the traffic for the maintenance of cached data items is not needed. However, this method cannot guarantee that their cached data items are true CNNs, and thus the accuracy of the query result is low as shown in Fig. 3.7(c). In the FA method with high periodic maintenance, the traffic increases as C increases, because with increasing C, more data items are exchanged at short intervals for data maintenance. The traffic in the FA method with low periodic maintenance is greater than that in the FA method with high periodic maintenance when k is less than 20, because the traffic involved in data maintenance is small, but that required by query processing becomes quite large.

From Fig. 3.7(b), the response time in all methods except the EXP method decreases as C increases. When C is more than 30, the response time in all the other methods is less than that of the EXP method. This is because when C is more than k, the query-issuing node can often acquire kNNs from its own cached data items. The response time in the FA method and the FA method by all nodes and with high periodic maintenance is nearly zero when C is more than 40, because with more frequent maintenance, there is a greater chance that query-issuing nodes will acquire kNNs from their own cached

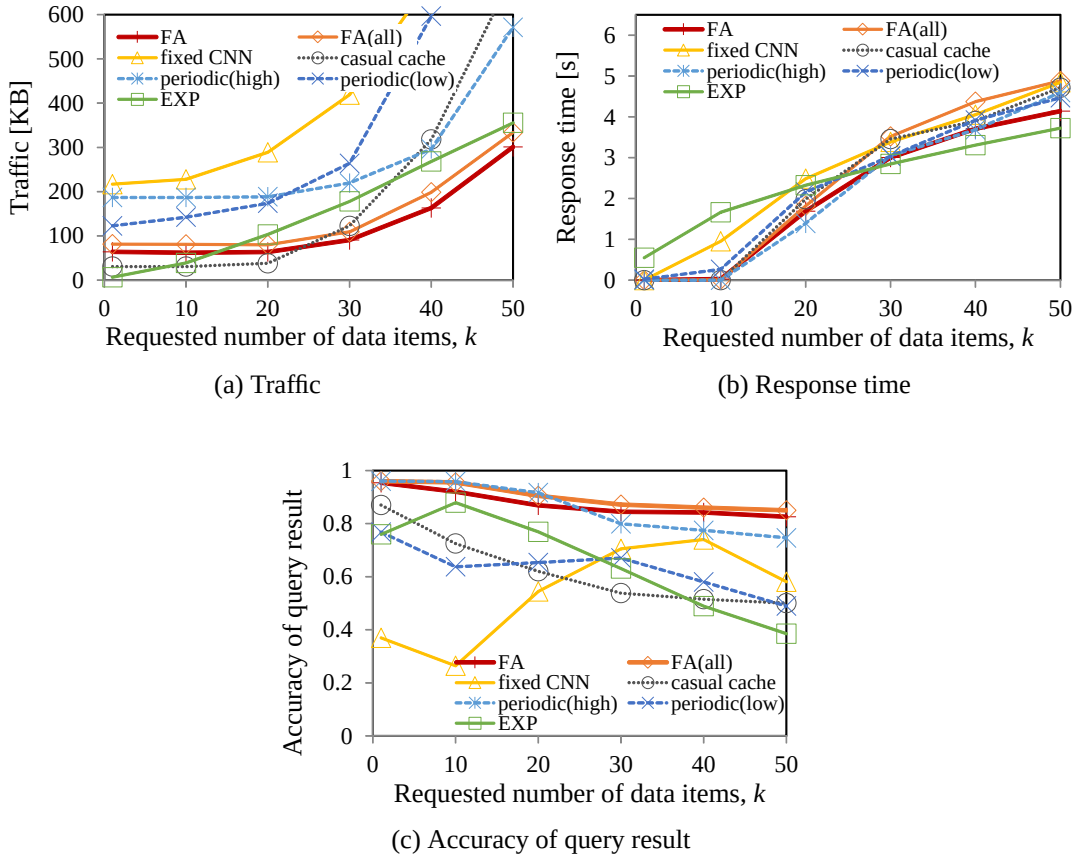
data items. In the FA method with casual caching, when C is more than 40, the response time is nearly zero, because query-issuing nodes nearly always acquire the kNN result from their own cached data items. However, this method cannot guarantee that their cached data items are true CNNs, and thus the accuracy of the query result is low as shown in Fig. 3.7(c).

From Fig. 3.7(c), the accuracy of the query result in the FA method and the FA method by all nodes is very high. This is because data maintenance in these methods is highly effective, and a smaller number of nodes send back replies to the query-issuing node. This also shows that the FA method can maintain high accuracy of the query result and also reduce traffic than the FA method by all nodes (shown in Fig. 3.7(a)). In the FA method with high periodic maintenance, when C is less than 20, the accuracy of the query result decreases because packet collisions often occur since more data items are exchanged at short intervals for data maintenance. In the FA method with fixed CNNs and low periodic maintenance, the accuracy of the query result is low because data items cached by respective nodes are typically far from the current C nearest data items; thus replies often do not cover the Filling Area and some necessary data items are lacking. When C is more than 30, the accuracy of the query result in the FA method with fixed CNNs is lower than in the FA method with low periodic maintenance because in the former, nodes' current locations are typically farther from their update points. In the FA method with casual caching, the accuracy of the query result is low, and increases as C increases. As the number of cached data items increases, query-issuing nodes incidentally retain kNNs more often in this method. In the EXP method, the accuracy of the query result is very low, because all nodes within the search area reply to a given query, which often causes message collision.

3.5.3 Impact of Requested Number of Data Items k

Fig. 3.8 shows the simulation results with varying k. In the graphs, the horizontal axis indicates the requested number of data items, k, and the vertical axes respectively indicate the traffic in Fig. 3.8(a), the response time in Fig. 3.8(b), and the accuracy of the query result in Fig. 3.8(c).

From Fig. 3.8(a), as k increases, the traffic increases in all methods, because the search area for processing a kNN query and the number of data items in replies increase.



(a) Traffic

(b) Response time

(c) Accuracy of query result

Figure 3.8: Impact of requested number of data items, k

In the FA method and the FA method by all nodes, the traffic is small and nearly constant when k is smaller than C, because query-issuing nodes can acquire all the kNNs from their own cache in most cases. The traffic in the FA method is smaller than in the FA method by all nodes. This is because in the FA method, the traffic for the maintenance of cached data items is smaller due to the effectiveness of MS mode. When k is small, the traffic in the FA method is relatively large in comparison with the FA method with casual caching and the EXP method, because the traffic for data maintenance becomes dominant. However, as k increases, the traffic in the FA method becomes less than in the other methods. The traffic in the FA method with fixed CNNs is substantial because of multiple replies and message flooding for notification of data generation. The traffic in the FA method with casual caching becomes greater as k increases because of multiple

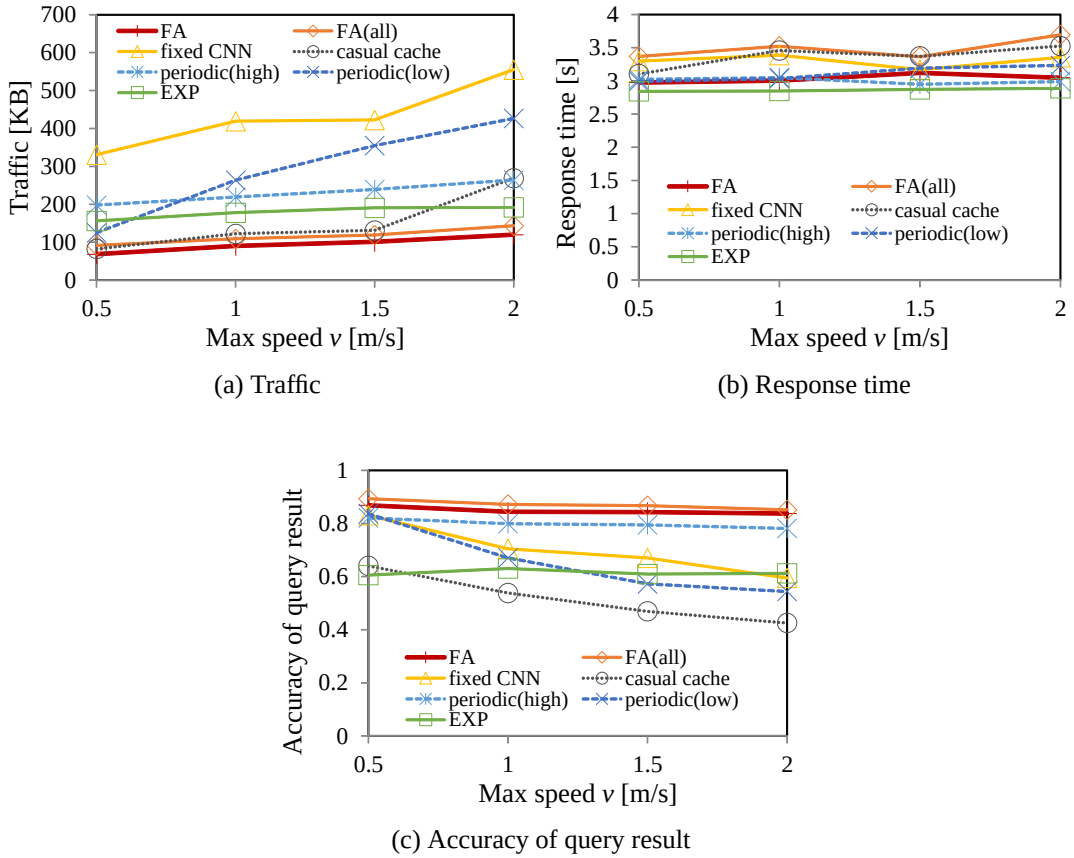
replies. In the FA method with high periodic maintenance, the traffic is greater than in the FA method, because the traffic involved in data maintenance is greater. The traffic in the FA method with low periodic maintenance is greater than that in the FA method with high periodic maintenance when k is more than 30, because the traffic involved in data maintenance is small, but that required by query processing becomes quite large with increasing k.

From Fig. 3.8(b), in all methods, the response time increases as k increases, because the response time depends on the size of the search area, which increases as k increases. In all the methods except the EXP method, the response time is very short when k is less than 10, because in most cases query-issuing nodes can acquire kNNs from their own cached data items.

From Fig. 3.8(c), in the FA method and the FA method by all nodes, the accuracy of the query result is high. As k increases, the accuracy of the query result in these methods decreases, because packet losses often occur due to a large number of replies. In the FA method with low periodic maintenance and that with fixed CNNs, the accuracy of the query result is low, because kNNs are often not cached on nodes within the search area. In the FA method with casual caching, the accuracy of the query result decreases as k increases, although the search area becomes larger. This is because each node wrongly assumes that it covers a wider area than the area covered by real CNNs, and thus nodes receiving a reply message often do not reply with data items since they misjudge that their assignment is fully covered by others. When k is 1, the accuracy of the query result in the EXP method is low, because the search area is set very small, and thus occasionally no data items whose data points are within the search area are present. Moreover, as k increases, the traffic significantly decreases because all nodes within the search area reply to a given query, which often causes message collision.

3.5.4 Impact of Node Mobility

Finally, we examine the impact of node speed. Fig. 3.9 shows the simulation results with varying maximum node speed. In the graphs, the horizontal axis indicates the maximum speed of nodes, v, and the vertical axes respectively indicate the traffic in Fig. 3.9(a), the response time in Fig. 3.9(b), and the accuracy of the query result in Fig. 3.9(c).



(a) Traffic

(b) Response time

(c) Accuracy of query result

Figure 3.9: Impact of node mobility

From Fig. 3.9(a), in all the methods, as v increases, the traffic slightly increases, because the data maintenance procedures occur more often. When v is more than 1, the traffic in the FA method with low periodic maintenance is greater than in the FA method with high periodic maintenance. This is because as v increases, the traffic involved in query processing increases, since nearby nodes cache less similar data items but remote nodes often cache the same data items, leading to multiple replies with the same data items. From Fig. 3.9(b), the response time in all the methods is nearly constant because the search area is nearly constant.

From Fig. 3.9(c), in the FA method and that by all nodes and with periodic maintenance, as v increases, the accuracy of the query result is nearly constant. In the FA method with fixed CNNs and with low periodic maintenance, as v increases, the ac-

curacy of the query result decreases, because many nodes move far from their update point, and thus, the Filling Area cannot be covered by cached data items of nodes near the query point. Therefore, we again have confirmed that data maintenance is important when nodes move rapidly. In the FA method with casual caching, the accuracy of the query result decreases as v increases. This is because each node wrongly assumes that it covers a wider area as v increases, and thus nodes receiving a reply message often do not reply with data items since they misjudge that their assignment is fully covered by others.

3.6 Conclusion

In this chapter, we have proposed a kNN query method; the Filling Area (FA) method for searching location-dependent data items, which aim at reducing traffic and maintaining high accuracy of the query result in MANETs. In the FA method, to achieve a small search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose locations are near their own. When a node issues a query, neighboring nodes send back their copies, which will be likely included the query result, with avoidance of duplicate data items being sent back. The experimental results show that the FA method reduces the traffic involved in processing kNN queries, and also achieves high accuracy of the query result.

In the FA method, the traffic for data maintenance is suppressed by canceling updating cached data items when a neighboring node has already cached same data items. However, it is more effective that nodes update their cached data items based on the query-issuing characteristics such as query frequency and query point. As part of our future work, we plan to extend our method by addressing this issue.

Chapter 4

Detecting the Convex Hull

4.1 Introduction

In an LBS, real-time location-specific queries for information held by mobile nodes are often used; and in such cases, convex hull queries, which retrieve the information necessary for calculating the convex hull of all the nodes composing a given network, are useful. By using convex hull queries, the query-issuing node can determine the convex hull of the network, which helps to know both the convex area in which nodes are present, and the farthest pair of nodes.

Numerous mathematical methods have been proposed for determining the convex hull of a set of points (node locations) [6]. In order to apply such algorithms to a MANET, however, the query-issuing node must first acquire the information on all node locations in the network. In a naïve approach, the query-issuing node can acquire the information on nodes composing the convex hull of the network by flooding a query throughout the entire network, and receiving location information on all the nodes. However, as described before, this approach produces exceedingly large traffic, and this leads to the loss of necessary data. Here, the convex hull can be calculated solely on the information on nodes which are vertices of the convex hull of the network. In other words, information on nodes which are not convex hull vertices is needless for calculating the convex hull. Therefore, it is efficient for the query-issuing node to acquire only the information on nodes which are vertices of the convex hull, since reducing query processing traffic is significant in MANETs.

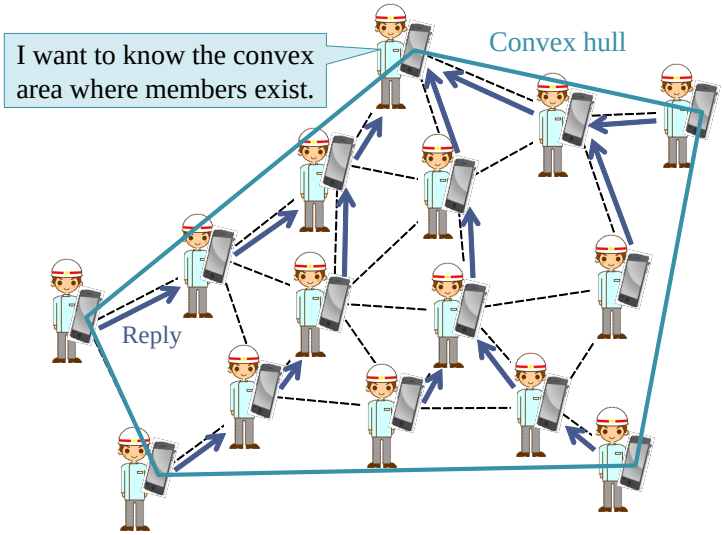


Figure 4.1: Example of performing a convex hull query in a MANET

Figure 4.1 shows an application example of a convex hull query in which a rescue worker attempts to determine the convex hull of nodes (rescue members) in a disaster site. If the query-issuing node receives the information on all nodes within the entire network, a substantial amount of unnecessary traffic is generated, and excessive traffic increases the probability of packet losses, which results in low accuracy of the query result.

In this section, we address the detection of the convex hull which is one type of boundaries. One of the differences between other boundaries proposed in existing methods is that the shape of convex hulls is uniquely defined regardless of network condition (*e.g.*, the size of nodes' communication range and the density of nodes). This is an advantage because users can search the network area without specifying any information to the convex hull query. In addition, the convex hull detection is useful to roughly know the size of networks. For example, before a node processes a kind of queries, it determines the parameters for the query processing based on the information acquired from the result of a convex hull query (*e.g.*, the maximum distance between nodes in the network). In such a case, the node does not need to know the shape of the network in details, and thus, the convex hull query is suitable for achieving it.

Therefore, we propose two convex hull query processing methods for MANETs, the Local Convex Hull (LCH) and Local Wrapping (LW) methods, for reducing traffic

while maintaining high accuracy of the query result. In these methods, nodes reply with information on nodes which are vertices of the convex hull. More specifically, in the LCH method, the query-issuing node first floods a convex hull query throughout the entire network. Then, each node replies with information on nodes which are vertices of the local convex hull, which is the convex hull composed of nodes whose information has already been received. In the LW method, to avoid transmitting queries throughout the entire network, the query-issuing node first sends a convex hull query to a node located on the outer boundary of the network, using a geo-routing technique. Then, the query is transmitted only to nodes on the outer boundary, until the query path forms a loop. In this way, unnecessary reply transmissions can be suppressed, even if mobile nodes do not know their neighbors in advance. We also show experimental results verifying that our proposed methods can reduce traffic compared with a naïve method, and also achieve high accuracy of the query result.

The remainder of this chapter is organized as follows. In Section 4.2, we introduce related work. In Section 4.3, we present our proposed convex hull query processing methods. In Section 4.4, we show the results of the simulation experiments. Finally, in Section 4.6, we summarize this chapter.

4.2 Related Work

In this section, we review some existing studies related to our work in this chapter.

4.2.1 Convex Hull Determination Algorithm

Numerous mathematical methods have been proposed for calculating the convex hull of a given set of points [6]. Here, we focus on two particularly relevant techniques, and discuss similarities and differences with respect to our approach. In [30], the author proposes a method for convex hull determination, called Jarvis's march, which is inspired by gift wrapping. In this method, the point with the smallest y-coordinate among all the points is first selected as a vertex of the convex hull. Then, the angle of each point with the x-axis is calculated, where the origin is fixed at the newly selected point, and the point with the smallest angle is newly selected as a vertex point of the convex hull. By

repeating this procedure, the method can determine the convex hull of a set of points. The LW method proposed in this chapter is inspired by this method.

In [24], the author proposes a method called the Graham scan. In this method, all points are first sorted based on their respective angles with the x-axis, where the origin is fixed at the point with the smallest y-coordinate among all the points. This method determines whether each point is a vertex of the convex hull, in the sorted order. These methods can determine convex hulls with low calculation costs when the positions of all points are already known; however, this is impossible in MANETs. Therefore, in this chapter, we aim to reduce the traffic required for acquiring the information on vertex nodes of the convex hull, while maintaining high accuracy of the result.

4.2.2 Boundary Detection in Various Networks

To the best of our knowledge, this thesis is the first to focus on convex hull query processing in MANETs. In this subsection, we review some existing works on boundary detection, which is closely related to convex hull determination [33, 67, 86, 88]. In [17], the authors propose a method for location-free boundary detection in MANETs. To determine the inner and outer boundaries without knowing node locations, node connectivity information is transferred to a geographical graph, and the global skeleton of the entire network is refined as the boundary. In [85], the authors propose a method for detecting the boundary of 3D sensor networks, based on nodes' neighborhood information. In this method, first triangular boundary surfaces are roughly constructed based on the Voronoi diagram, and then, certain nodes located outside these surfaces are identified as the final boundary nodes. In [56], the authors propose a method for detecting the geographical boundary of sensor readings in a dense sensor network. In this method, node send their sensor readings to a sink only when they judge that the readings contribute to the boundary detection by overhearing the sensor readings of neighboring nodes. These methods determine various types of boundary, however in this chapter we aim to determine the geographic convex hull, which is different, both in focus and approach, from these existing works.

4.3 Convex Hull Query Processing Methods

In this section, we describe how convex hull queries are processed in our LCH and LW methods. The system environment is same as described in Section 2.3.

4.3.1 Local Convex Hull (LCH) Method

When nodes reply after a query is transmitted throughout the network, relay nodes can identify the nodes which have no possibility of being vertices of the convex hull, by calculating the local convex hull based on received node information. The LCH method reduces unnecessary traffic because the information on nodes which cannot be vertex nodes of the convex hull is not sent back in reply.

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

1. M_s transmits a convex hull query message to its neighboring nodes. In the query message, the query-issuing node's ID and location are set as M_s and its location, respectively.

2. Each mobile node, M_p, on first receiving the query message, stores the identifier of the source node as its parent. It sets the waiting time T for sending a reply, according the following equation:

$$T = \alpha \left(1 - \frac{a}{d}\right), \tag{4.1}$$

where a is the distance between M_s and M_p, d ($>a$) is the maximum distance (a positive constant), and α is a parameter specified by the system designer to avoid message collision. As Equation (4.1) shows, T decreases as the distance between M_s and M_p increases.

At the same time (without waiting T), M_p broadcasts the query message to its neighboring mobile nodes.

If M_p has already received the query message, it discards the message and does nothing.

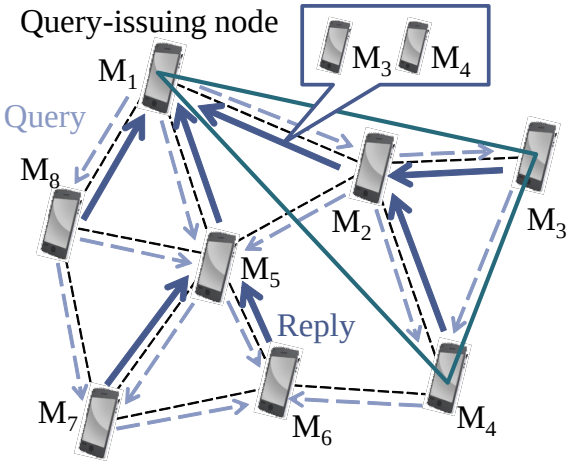


Figure 4.2: Convex hull query processing in the LCH method

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

4. Each node that receives the reply message updates the tentative convex hull result included in the reply message, by adding the information on itself and calculating the new convex hull by using some existing method (such as [24]).

5. When its T has passed, if the node is not the query-issuing node, it transmits a reply message, including the updated tentative convex hull result, to its parent, and the procedure returns to Step 4. Otherwise (*i.e.*, the node is the query-issuing node), the query processing is completed.

Figure 4.2 shows an example of executing the LCH method, where M_1 is the query-issuing node. When M_2, M_5, and M_8 receive a query message from M_1, they store M_1's ID as their parent, and broadcast the query message to their respective neighboring nodes. When T has passed at M_4 (M_3), M_4 (M_3) transmits a reply message, including M_4 (M_3)'s information, to M_2. On receiving the reply messages from M_3 and M_4, M_2 calculates its local convex hull based on the information on M_1–M_4. When T has passed at M_2, M_2 transmits a reply message, including only M_3's and M_4's information, because M_2 is located inside the local convex hull. Such a procedure is performed

throughout the entire MANET, until finally M_1 acquires the information on all nodes in the network.

4.3.2 Local Wrapping (LW) Method

In the LCH method, a large volume of unnecessary traffic may occur because queries are transmitted to all nodes throughout the network. In order to effectively reduce traffic, queries should be transmitted only to nodes which are included in the query result (*i.e.*, vertex nodes of the convex hull). Given the geographic characteristics of the convex hull, the vertex nodes of the convex hull basically define the network edge (outer boundary). Therefore, we propose the LW method, which does not require query flooding throughout the network.

In this method, a query is transmitted to nodes on the outer boundary of the network. More specifically, the query-issuing node first sends a query to the node with the smallest y-coordinate (*start node*), and then the query is forwarded along the outer boundary from this start node. Figure 4.3 shows an example of executing the LW method, where M_5 is the query-issuing node. In this example, the query is first transmitted to M_7, which has the smallest y-coordinate, and then is forwarded, counterclockwise, to nodes enclosing the network (*i.e.*, M_6, M_4, M_3, M_2, M_1, M_8, and M_7).

To effectively forward a query through the outer nodes, in the LW method, each node receiving the query determines the next node to which the query should be sent, in a three-way handshake manner. Specifically, it first sends a search message to its neighbors, and each node receiving this message sets the waiting time for reply based on the angle between the vertex nodes whose information has already been acquired. When the node with the smallest angle firstly replies, the query-sending node sends the query to this node as the next node.

In this section, we first describe the geo-routing method for forwarding a query to the start node, and then describe how convex hull queries are processed in the LW method.

Geo-routing Method for Forwarding a Query to the Start Node

In the LW method, the query-issuing node first forwards a convex hull query, using our proposed geo-routing method in Section 2.4.1, to the start node, which has the smallest y-coordinate. Our geo-routing method adopts a three-way handshake protocol to send a

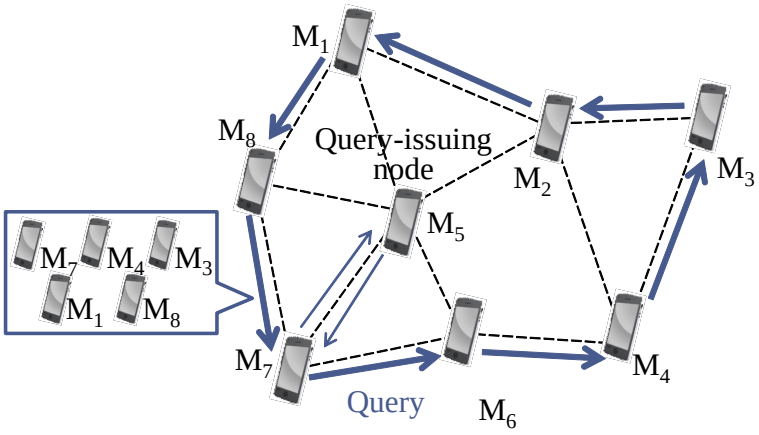


Figure 4.3: Convex hull query processing in the LW method

query to the neighboring node closest to the target point among its neighboring nodes. By repeating this procedure, the query is forwarded to the node closest to the target point.

More specifically, in the geo-routing method, the query-issuing node first broadcasts a search message. Then, when a node receives the search message, if it is closer to the target point than the source node, it sets the waiting time for sending a reply. Because nodes closer to the target point, transmit a reply message after a shorter waiting time, the node closest to the target point, among the neighbors, is the first to transmit a reply message to the source node. Then the source node, which has now received reply messages from its neighbors, sends a (forwards the) convex hull query message only to the node that first sent the reply. The node receiving the forwarded convex hull query broadcasts a search message using the same procedure. Finally, if a node that has sent a search message does not receive any reply messages, it recognizes itself as the node closest to the target point (*i.e.*, the start node).

Through the above procedure, the query-issuing node can transmit the query to the start node with low traffic, because the geo-routing method requires neither the use of beacon messages nor the construction of multiple paths.

Algorithm 3 : Tentative result updating

1: D : tentative result
2: i = # of data in D
3: $j = i - 1$
4: **while** $\angle$ D[i] D[j] D[$j-1$] $< \pi$ **do**
5: remove D[j] from D
6: $j = j - 1$
7: **end while**

Forwarding a Convex Hull Query and Returning the Result

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

1. M_s transmits a convex hull query to the start node, which has the smallest y-coordinate (one of the nodes on the outer boundary), using the geo-routing method described in Section 4.3.2. In the geo-routing method, the target point is set to $(x, y - r)$, where the source node is located at (x, y). In the query message, the query-issuing node's ID and location are set as M_s and its location, respectively. When a node receives no reply message, it recognizes itself as the start node, and begins acquiring the desired information.

2. After receiving the query (forwarding) message, M_t (*the query holder*) updates the *tentative result* by adding its ID and location. Note that the first query holder is the start node. When the number of node IDs in the tentative result is greater than 3, the node determines the vertex nodes of the convex hull in the tentative result, using Algorithm 3. Through the use of this algorithm, the information on nodes which cannot be vertex nodes of the global convex hull is omitted from the tentative result.

3. M_t broadcasts a neighbor search message to its neighboring mobile nodes. The neighbor search message includes the information (ID and location) on the two nodes (M_t and M_q) last added to the tentative result. Here, if M_t is the start node, the reply message includes only the ID and location of M_t.

4. Each mobile node, M_p, that has received the neighbor search message sets the waiting time T for sending a *response message* according to the following equation:

$$T = \frac{t_m \theta}{2\pi}, \tag{4.2}$$

where t_m is a positive constant specifying the maximum time before sending a response message, and θ is the angle of $\angle\ M_q\ M_t\ M_p$. If the information on M_q is not included in the neighbor search message (*i.e.*, the message is sent from the start node), θ is calculated as the angle with the x-axis where the origin is fixed at M_t. When M_q=M_p, θ is 2π; *i.e.*, T = t_m. As Equation (4.2) shows, T decreases as the angle of $\angle\ M_q\ M_t\ M_p$ decreases.

5. When T has passed at M_p, the node transmits a response message, including its own ID, to the sender of the neighbor search message, M_t.

6. When M_t receives a response message for the first time, if it is the start node (*i.e.*, the query path forms a loop), it initiates a reply to the query-issuing node, according to Step 8. Otherwise, M_t transmits a *query forwarding message* to the sender of the response message. This message includes the query-issuing node's ID (M_s) and location, and the tentative result. Meanwhile, the nodes that overhear the response message stop sending their response message.

7. The node, M_p, which receives the query forwarding message, becomes the new query holder, and the procedure returns to Step 2.

8. The start node replies, with the tentative result as the final result, to the query-issuing node, using the geo-routing method described in Section 4.3.2, where the target point is set to the location of the query-issuing node.

Figure 4.4 shows an example of executing the LW method, where M_6 receives a query forwarding message from M_7. First, M_6 broadcasts a neighbor search message to its neighboring nodes. When M_4 (M_5, M_7) receives the message from M_6, it sets T based on $\angle\ M_7\ M_6\ M_{4(5,7)}$ (see Figure 4.5). Since M_4 has set the smallest T, because $\angle\ M_7\ M_6\ M_4 < \angle\ M_7\ M_6\ M_{5(7)}$, M_4 is first to transmit a response message to M_6, when T has passed. When M_6 receives the response message from M_4, it transmits a

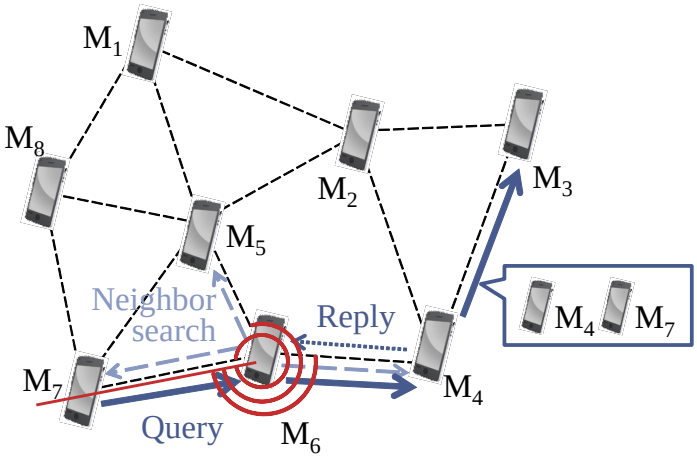


Figure 4.4: Example of query forwarding in the LW method

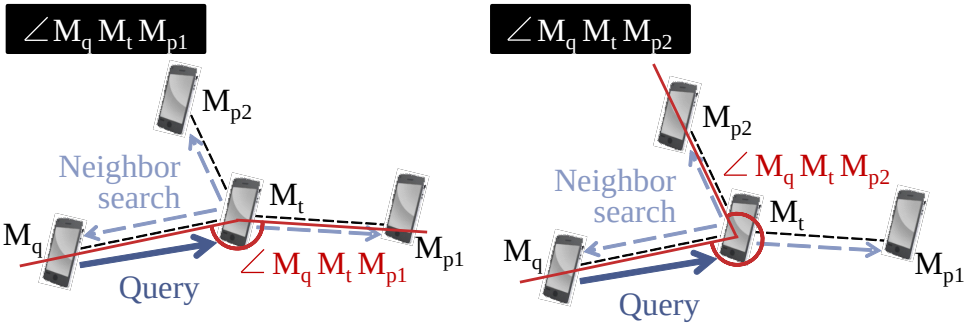


Figure 4.5: Example of angles for setting T

query forwarding message, with the information on M_7 and itself, to M_4. Here, M_5 and M_7 (both of which have received the response message) stop sending their response message. When M_4 receives the query forwarding message from M_6, it becomes the query holder, and transmits a neighbor search message. At the same time, it updates the tentative result, which includes the information on M_4 and M_7, because M_6 is not a vertex node of the convex hull since $\angle\ M_4\ M_6\ M_7 < \pi$. These procedures are repeated until the query is transmitted to the start node.

The LW method can acquire the information on the convex hull vertex nodes of a given network with lower traffic than the LCH method, because here queries are forwarded only to the boundary nodes of the entire network. It is also expected that the

response time would be reduced, because nodes select subsequent forwarding nodes by setting T based on the respective angles, without acquiring the information on all neighboring nodes.

4.4 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 QualNet5.2[66].

4.4.1 Simulation Model

The number of mobile nodes in the entire system is n (M_1, $\cdots$, M_n). These mobile nodes are present in an area of 1,000[m] $\times$ 1,000[m], and move according to the random walk model [8], with a movement speed randomly determined from 0.1 to v[m/sec]. 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 above setting on the network is basically same as that in Section 2.5.1. α in Equation (4.1) and t_m in Equation (4.2) are respectively set as (1.0 + n/250) and 0.07, based on our preliminary experiments.

We compare the performance of our proposed LCH and LW methods with that of a naïve method. The node behavior in the naïve method is similar to that in the LCH method, except that, in the former, relay nodes transmit the information on all the nodes (*i.e.*, without calculating the local convex hull). In the naïve method, α is set to either (1.0 + n/250) or 2(1.0 + n/250), to evaluate the impact of the waiting time for reply. In the graphs, the naïve method with these α values is denoted as 'Naïve' and 'Naïve (double)', respectively.

In the above simulation model, we randomly determine the initial position of each mobile node. The query-issuing node, randomly chosen among all nodes, issues a convex hull query. We repeat this process 200 times (*i.e.*, 200 queries) for every 10 seconds, and evaluate the following four criteria.

- Traffic

  We examine the total volume of query messages and replies exchanged in processing a query. Table 4.1 shows the size of messages used in our methods and the naïve method, where p denotes the number of nodes whose information is included in the reply. We define 'traffic' as the average of the total volume of respective queries issued.

- Response time

  We examine the time from the transmission of a query message by the query-issuing node, to the reception of the result. In the naïve 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 vertex nodes whose information is included in the result acquired by the query-issuing node, to that of all the vertex nodes of the convex hull (*i.e.*, the complete answer). We define 'accuracy of query result' as the average of the query result for all queries issued.

- Area accuracy

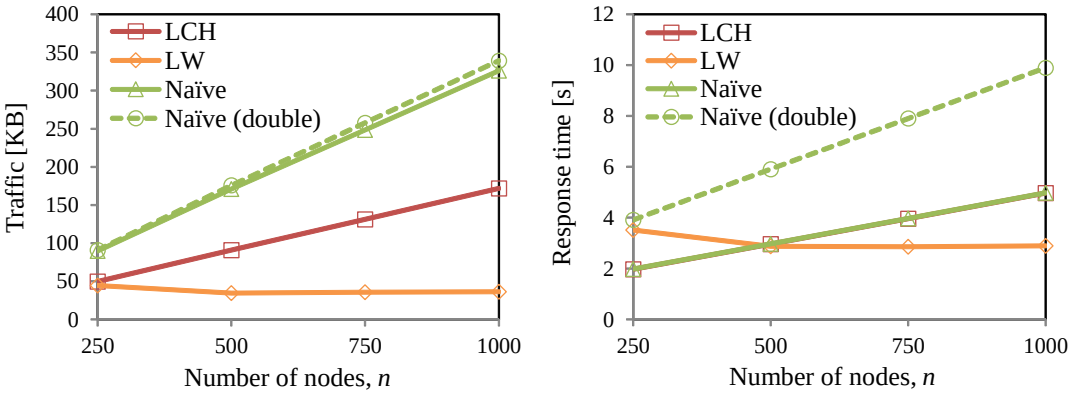
  We examine the ratio of the area of the convex hull of nodes whose information is included in the result acquired by the query-issuing node, to the area of the real convex hull of nodes (*i.e.*, the complete answer). We define 'ratio of calculated area' as the average of such an area for all queries issued.

4.4.2 Impact of node number n

First, we examine the performance of our proposed methods with varying number of nodes n. Figure 4.6 shows the simulation results. In these graphs, the horizontal axis indicates the number of nodes n, and the vertical axis indicates the traffic in Figure

Table 4.1: Message types and sizes

Type	Size [B]
Query	64
Neighbor search (LW method)	48
Response (LW method)	16
Query forwarding (LW method)	64+32p
Reply	64+32p
Ack to the received reply	16



(a) Traffic (b) Response time

(c) Accuracy of query result (d) Area accuracy

Figure 4.6: Impact of node number n

4.6(a), the response time in Figure 4.6(b), the accuracy of the query result in Figure 4.6(c), and the area accuracy in Figure 4.6(d).

From Figure 4.6(a), it can be seen that our proposed methods significantly reduce traffic compared with the naïve method. Of the two methods, the LW method generates the lowest traffic. These results show the effectiveness of our methods. As n increases, the traffic in the LCH and naïve methods increases. This is because, in these methods, a query is sent to all nodes in the network. In addition, in the naïve method all nodes reply with their location information, which results in significant traffic, especially when the number of nodes is large. In contrast, in the LW method, even if n increases, the traffic is almost constant. This is because the query is transmitted only to nodes which are on the outer boundary, and thus the traffic basically depends on the number of hops in the query path, not on the number of nodes in the network. When n is 250 (*i.e.*, the density of nodes is low), the traffic is slightly higher since the chances of message retransmission increase due to frequent link disconnection.

From Figure 4.6(b), as n increases, the response time in the LCH and naïve methods increases. This is because the waiting time for reply is set based on the number of nodes, to avoid reply message collisions. In contrast, in the LW method the response time is almost constant regardless of n. This is due to the same reason as described above. As a result, the LW method results in the shortest response time when the number of nodes is large.

From Figure 4.6(c), the accuracy of the query result in the LCH method is very high, while that in the LW method is slightly lower. This is because, in the LW method, the accuracy of the query result decreases due to frequent link disconnections during query transmission. Especially when n is 250 (*i.e.*, the density of nodes is low), the accuracy of the query result decreases because the chances of link disconnection increase. The accuracy of the query result in the naïve method is lower than in our proposed methods. This is because message collisions often occur, due to the large size of reply messages. When n is large, the accuracy of the query result is especially low, as message collisions occur more frequently. Here, if the query-issuing node can accept a long waiting time for acquiring the result (*i.e.*, 'Naïve (double)' in the graph), the accuracy of the query result increases because the chances of collision decrease; however, in this case the response time increases, of course, as shown in Figure 4.6(b).

From Figure 4.6(d), the area accuracy is very high in our proposed methods. In the

LW method, when n is 250, the area accuracy slightly decreases (*i.e.*, the query-issuing node determines a smaller convex hull than the actual one). This is because a vertex node is sometimes geographically isolated due to low node density, and thus the query-issuing node cannot acquire the information on it. In contrast, in the naïve method, as n increases, the area accuracy typically decreases due to more frequent packet losses, which results in a significant difference in area accuracy between the naïve method and those proposed here.

4.4.3 Impact of node speed v

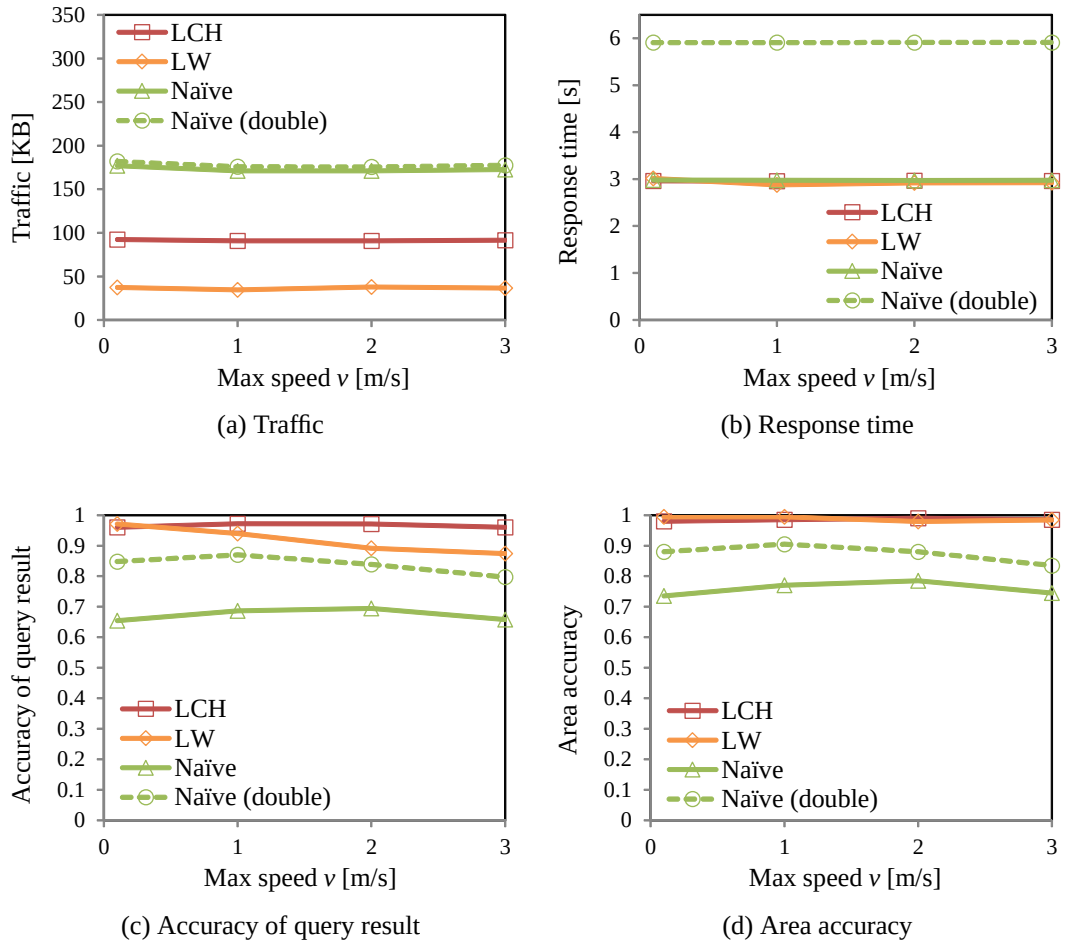
We examine the performance of our proposed methods with varying node speed. Figure 4.7 shows the simulation results. In these graphs, the horizontal axis indicates the maximum speed of nodes, v, and the vertical axis indicates the traffic in Figure 4.7(a), the response time in Figure 4.7(b), the accuracy of the query result in Figure 4.7(c), and the area accuracy in Figure 4.7(d).

From Figure 4.7(a), the traffic is nearly constant, regardless of the nodes' speed, in all the methods. Similar to the results in Figure 4.6(a), our methods produce significantly lower traffic than the naïve method.

From Figure 4.7(b), the response time is also nearly constant in all the methods. This is because, in the LCH and naïve methods, the response time depends on α, whereas in the LW method it depends on t_m.

From Figure 4.7(c), the accuracy of the query result in the LCH and naïve methods is nearly constant, regardless of v. Especially in the LCH method, the accuracy of the query result is very high, even when v is large. On the other hand, in the LW method, the accuracy of the query result, though still high, decreases as v increases. This is because this method requires more time to forward query messages than the other methods, and thus there is a greater probability that the set of vertex nodes of the convex hull will change during query transmission.

From Figure 4.7(d), the area accuracy is nearly 1 in our proposed methods. The LW method achieves an area accuracy of nearly 1, even when v is large, though the accuracy of the query result is slightly lower, as shown in Figure 4.7(c). This shows that even if the set of vertex nodes of the convex hull changes due to node movement, which decreases the accuracy of the query result, the geographical area of the convex



(a) Traffic (b) Response time

(c) Accuracy of query result (d) Area accuracy

Figure 4.7: Impact of node speed v

hull remains almost same, meaning that the query-issuing node can relatively accurately ascertain the area of the convex hull.

4.5 Convex Hull Detection for Location-Dependent Data

As we described in Chapter 3, the search targets of queries are not only nodes themselves but also data items which nodes hold. When a node wants to know a convex hull of data items which satisfy some conditions which the node specifies (*e.g.*, the score of data items is over 40), it is useful to use convex hull queries in a MANET. Therefore,

we here discuss convex hull queries for data items.

There are two types of convex hulls for data items. One type is the convex area where all data items which satisfy the specified condition are associated (*i.e.*, the search target is associated locations of data items). For example, a node can know the area where an infectious disease occurs by knowing data items on the patients with the disease. The query-issuing nodes can easily detect this type of convex hulls by using the LCH method.

The other type of convex hulls is the area where there exist all nodes which have data items which satisfy the specified condition (*i.e.*, the search target is locations of nodes). It is useful to detect such an area because nodes can use the convex hull detection as the first step for acquiring necessary data items, since nodes can acquire necessary data items by searching the detected area. If neighboring nodes have the target data items, the convex hull enclosing all requested data items becomes small, which makes small traffic and time for searching. Therefore, efficient data management, *i.e.*, placement of data items and copies, is key challenges. For this aim, the method proposed in [65] can be used with some modification. In this method, a region is divided into some areas, and the nodes within each area determine data items to allocate based on the scores of data items. By doing so, the query-issuing nodes can acquire necessary data items without flooding within the entire network.

As mentioned above, convex hull query processing for data items in MANETs can be processed by using some existing methods, and we omit the detail of how to achieve this.

4.6 Conclusion

In this chapter, we have proposed two methods for processing convex hull queries in MANETs, which aim to reduce traffic and response time while maintaining high accuracy of the query result. In these methods, nodes reply with information on nodes which are vertices of the convex hull composed of nodes whose information has already been acquired. In the LCH method, the query-issuing node first floods a convex hull query throughout the entire network. Then, each node replies with information on nodes which are the vertices of the local convex hull, which is a convex hull composed

of nodes whose information has already been received. In the LW method, the query-issuing node first sends a convex hull query to a node located on the outer boundary of the network, using a geo-routing technique. Then, the query is transmitted only to nodes on the outer boundary, until the query path forms a loop. In this way, the convex hull can be determined without acquiring the information on all nodes in the network.

The experimental results show that our proposed methods reduce the traffic and response time in comparison with the naïve method, and also achieve high accuracy of the query result. More specifically, the LCH method achieves high accuracy of the query result regardless of the number of nodes and node speed. On the other hand, the LW method achieves the best performance when the number of nodes is large, because queries are transmitted only to nodes with a high probability of being included in the query result.

In the LCH method here described, though the accuracy of the query result is high, the traffic becomes large when the number of nodes is large, because all nodes relay a query. We thus plan to extend the LCH method, to reduce the traffic required for query transmission, while ensuring the reception of the query by all nodes.

Chapter 5

Summary

5.1 Summary of Contributions

In this thesis, we have discussed query processing methods for location-based services in MANETs. In Chapter 1, we presented the importance of location-based services (*i.e.*, kNN and convex hull query processing) in MANETs and made clear research issues for achieving efficient query processing.

In Chapter 2, we have proposed two beacon-less query processing methods for searching k nearest neighbor nodes in MANETs. In these 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 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 kNN result transmits the result to the query-issuing node. The experimental results have shown that our proposed methods reduce traffic for query processing and achieve high accuracy of the query result, in comparison with existing methods.

In Chapter 3, we have focused on query processing for retrieving the k nearest

location-dependent data items from the location of the query issuer in MANETs. We have proposed the Filling Area (FA) method, which aims at reducing traffic and maintaining high accuracy of the query result. In the FA method, to achieve a small search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose locations are near their own. When a node issues a query, neighboring nodes send back their copies, which will be likely included the query result, with avoidance of duplicate data items being sent back. The experimental results have shown that the FA method reduces the traffic involved in processing kNN queries, and also achieves high accuracy of the query result.

In Chapter 4, we have proposed two convex hull query processing methods for reducing traffic while maintaining high accuracy of the query result in MANETs. By using convex hull queries, the query-issuing node can determine the convex hull of the network, which helps to know both the convex area in which nodes are present, and the farthest pair of nodes. In our proposed methods, nodes reply with information on nodes which are convex hull vertices. In the Local Convex Hull (LCH) method, nodes reply with information on nodes which are vertices of the local convex hull, which is the convex hull composed of nodes whose information has already been received. In the Local Wrapping (LW) method, the query is transmitted only to nodes on the outer boundary. The experimental results have shown that our proposed methods reduce traffic and high accuracy of the query result, in comparison with a naïve method. In addition, we have discussed convex hull query processing for location-dependent data.

In summary, our proposed methods can reduce traffic for query processing, and also achieve high accuracy of the query results for location-based services in MANETs. Especially, we have designed our methods as beacon-less methods (*i.e.*, nodes do not require exchanging messages including node information), which are suitable for MANETs. This achievement will contribute to increase the availability of MANET services.

5.2 Future Work

Through this thesis work, we found the following remaining issues open to our future work.

5.2.1 Short Response time

In MANETs, location-specific information (*e.g.*, kNNs from a given query point and vertex nodes of a given convex hull) may change during query processing, because nodes move freely; and when the network topology changes during query execution, the query-issuing node may not be able to acquire fully accurate query answer. To avoid this, we have designed our methods to execute a query using only one round of message transmissions in order to acquire the query result in a short time. The experimental results have also shown that the response time in our methods proposed in this thesis is around a few seconds, which is enough small while achieving high accuracy of the query result.

However, in a real environment, users require to acquire the query results in a short period of time as possible, especially, when a network becomes larger and the response time for query processing tends to be longer. Therefore, we plan to extend our methods for minimizing response time.

5.2.2 Continuous Query Processing

In this thesis, we have assumed that queries attempt to acquire kNNs and determine the convex hull only once (*i.e.*, snapshot queries). However, in a real environment, the query-issuing node may wish to continuously monitor kNNs or the convex hull of the entire MANET, since location-specific information may change due to node mobility, and new generation (insertion), update, and delete of location-dependent data items [10, 11, 13, 14, 22, 68, 80, 81, 84].

In a naïve approach, the query-issuing node repeatedly send queries over the network. However, even if the query-issuing node frequently processes queries to accurately know the changing answer sets, it may not be able to acquire fully accurate query results because the change may occur during the interval of query processing, although the traffic for sending query and reply messages increases. We thus plan to extend our proposed methods to enable continuous monitoring of the kNNs and convex hull (*i.e.*, continuous query processing).

Acknowledgment

Upon the completion of this thesis, I would like to take this opportunity to express my sincerest gratitude to those who have done their best to offer me assistances to the thesis.

First and foremost, I am deeply indebted to my honors supervisor, Professor Takahiro Hara, who not only accepted me as his student but also assisted me to overcome the limitation of my knowledge and converting me into a researcher in this novel research field, as well as providing me valuable advice and guidance throughout my undergraduate, master, and doctoral research work. This thesis would never have been completed without his unfailingly wise advice and support.

I would also like to acknowledge the committee members of my thesis, Professor Makoto Onizuka, and Professor Yasuyuki Matsushita at the Department of Multimedia Engineering of the Graduate School of Information Science and Technology of Osaka University. Their insightful and constructive comments considerably improved the quality of the thesis.

I would like to express my appreciation to Professor Toru Fujiwara, and Professor Shinji Shimojo at the Department of Multimedia Engineering of the Graduate School of Information Science and Technology of Osaka University.

I would also like to express my gratitude to President Shojiro Nishio at Osaka University, and Dr. Yuya Sasaki at Nagoya University for his continuous support and guidance throughout my undergraduate, master, and doctoral research work.

I am very grateful for the help and support from Professor Daichi Amagata, Professor Tomoki Yoshihisa, Professor Masumi Shirakawa, Professor Akimitsu Kanzaki at Shimane University, Dr. Yuich Teranishi at National Institute of Information and Communications Technology, Professor Kaname Harumoto at Osaka University, and Professor Tadashi Nakano at Osaka University.

I would also like to thank my team members, Mr. Keisuke Goto, Mr. Kamalas Udomlamlert, Mr. Masahiro Yokoyama, Mr. Yuki Nakayama, Mr. Boqi Gao, Mr. Naoya Taguchi, Mr. Syunya Nishio, Mr. Keisuke Nakashima, and Ms. Liu Siqi for discussion of my thesis.

I sincerely appreciate Mr. Toshimitsu Fujii at SCSK Corporation, Mr. Takuji Tsuda at Mitsubishi Electric Corporation, and Mr. Duong Hong Nguyen for discussion of my research work.

With pleasure of studying and working with a number of very talented and warm hearted people at Hara laboratory, I would like to express thanks to them.

Last but by no means least it gives me immense pleasure to offer my hearty thanks to my family. No words can express my appreciation their support and love in my life. My parents have provided me with the greatest concerns and cares, for which I am eternally grateful.

Finally, with all my best wishes to those wonderful people who ever gave me assistances, supports and encouragements throughout my research experience.

REFERENCE

[1] Akkaya, K., and Younis, M.: A survey on routing protocols for wireless sensor networks, *Ad hoc networks*, Vol. 3, No. 3, pp. 325–349 (2005).

[2] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.: A survey on sensor networks, *IEEE Communications magazine*, Vol. 40, No. 8, pp. 102–114 (2002).

[3] Amagata, D., Sasaki, Y., Hara, T., and Nishio, S.: A robust routing method for top-k queries in mobile ad hoc networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 251–256 (2013).

[4] Baker, D. J., and Wieselthier, J. E.: A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network, in *Proc. of Int'l Conf. on Communications (ICC)* (1982).

[5] Bao, J., Chow, C.-Y., Mokbel, M. F., and Ku, W.-S.: Efficient evaluation of k-range nearest neighbor queries in road networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 115–124 (2010).

[6] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H.: The quickhull algorithm for convex hulls, *ACM Trans. on Mathematical Software (TOMS)*, Vol. 22, No. 4, pp. 469–483 (1996).

[7] Broch, J., Maltz, D. A., Johnson, D. B., Hu, Y.-C., and Jetcheva, J.: A performance comparison of multi-hop wireless ad hoc network routing protocols, in *Proc. of Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pp. 85–97 (1998).

[8] Camp, T., Boleng, J., and Davies, V.: A survey of mobility models for ad hoc network research, *Wireless communications and mobile computing*, Vol. 2, No. 5, pp. 483–502 (2002).

[9] Camp, T., Boleng, J., and Wilcox, L.: Location information services in mobile ad hoc networks, in *Proc. of Int'l Conf. on Communications (ICC)*, Vol. 5, pp. 3318–3324IEEE (2002).

[10] Chatzimilioudis, G., Zeinalipour-Yazti, D., Lee, W.-C., and Dikaiakos, M. D.: Continuous all k-nearest-neighbor querying in smartphone networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 79–88 (2012).

[11] Cheema, M. A., Brankovic, L., Lin, X., Zhang, W., and Wang, W.: Multi-guarded safe zone: An effective technique to monitor moving circular range queries, in *Proc. of Int'l Conf. on Data Engineering (ICDE)*, pp. 189–200 (2010).

[12] Chlamtac, I., Conti, M., and Liu, J. J.-N.: Mobile ad hoc networking: Imperatives and challenges, *Ad hoc networks*, Vol. 1, No. 1, pp. 13–64 (2003).

[13] Chon, H. D., Agrawal, D., and El Abbadi, A.: Range and k NN query processing for moving objects in grid model, *Mobile Networks and Applications*, Vol. 8, No. 4, pp. 401–412 (2003).

[14] Chow, C.-Y., Mokbel, M. F., and Leong, H. V.: On efficient and scalable support of continuous queries in mobile peer-to-peer environments, *IEEE Trans. on Mobile Computing (TMC)*, Vol. 10, No. 10, pp. 1473–1487 (2011).

[15] Cruz, L. A., Nascimento, M. A., and de Macêdo, J. A.: K-nearest neighbors queries in time-dependent road networks, *Journal of Information and Data Management*, Vol. 3, No. 3, p. 211 (2012).

[16] Demiryurek, U., Banaei-Kashani, F., and Shahabi, C.: Efficient k-nearest neighbor search in time-dependent spatial networks, in *Proc. of Int'l Conf. on Database and Expert Systems Applications (DEXA)*, pp. 432–449 (2010).

[17] Dong, D., Liu, Y., and Liao, X.: Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods, in *Proc. of Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 135–144 (2009).

[18] Ference, G., Lee, W.-C., Jung, H.-J., and Yang, D.-N.: Spatial search for K diverse-near neighbors, in *Proc. of Int'l Conf. on Information and Knowledge Management (CIKM)*, pp. 19–28 (2013).

[19] Fu, T.-Y., Peng, W.-C., and Lee, W.-C.: Parallelizing itinerary-based KNN query processing in wireless sensor networks, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 22, No. 5, pp. 711–729 (2010).

[20] Fujii, T., Kaji, M., Sasaki, Y., Hara, T., and Nishio, S.: A flooding control method with ack-carry for location-based information dissemination in mobile ad hoc networks, in *Proc. of Int'l Symposium on Applications and the Internet (SAINT)*, pp. 128–135 (2011).

[21] Gao, Y., Zheng, B., Chen, G., Lee, W.-C., Lee, K. C., and Li, Q.: Visible reverse k-nearest neighbor query processing in spatial databases, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 21, No. 9, pp. 1314–1327 (2009).

[22] Gao, Y., Zheng, B., Chen, G., and Li, Q.: Algorithms for constrained k-nearest neighbor queries over moving object trajectories, *Geoinformatica*, Vol. 14, No. 2, pp. 241–276 (2010).

[23] Goldsmith, A. J., and Wicker, S. B.: Design challenges for energy-constrained ad hoc wireless networks, *IEEE Wireless Communications*, Vol. 9, No. 4, pp. 8–27 (2002).

[24] Graham, R. L.: An efficient algorithm for determining the convex hull of a finite planar set, *Information Processing Letters*, Vol. 1, No. 4, pp. 132–133 (1972).

[25] Hagihara, R., Shinohara, M., Hara, T., and Nishio, S.: A message processing method for top-k query for traffic reduction in ad hoc networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 11–20 (2009).

[26] Hamida, E. B., and Chelius, G.: A line-based data dissemination protocol for wireless sensor networks with mobile sink, in *Proc. of Int'l Conf. on Communications (ICC)*, pp. 2201–2205 (2008).

[27] Hara, T., and Madria, S. K.: Consistency management strategies for data replication in mobile ad hoc networks, *IEEE Trans. on Mobile Computing*, Vol. 8, No. 7, pp. 950–967 (2009).

[28] Heissenbüttel, M., Braun, T., Bernoulli, T., and WäLchli, M.: BLR: Beaconless routing algorithm for mobile ad hoc networks, *Computer Communications*, Vol. 27, No. 11, pp. 1076–1086 (2004).

[29] IEEE 802.11, The Working Group for Wireless LANs: http://grouper.ieee.org/groups/802/11/.

[30] Jarvis, R. A.: On the identification of the convex hull of a finite set of points in the plane, *Information Processing Letters*, Vol. 2, No. 1, pp. 18–21 (1973).

[31] Jayaraman, P. P., Zaslavsky, A., and Delsing, J.: Cost-efficient data collection approach using k-nearest neighbors in a 3D sensor network, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 183–188 (2010).

[32] Jensen, C. S., Kolářvr, J., Pedersen, T. B., and Timko, I.: Nearest neighbor queries in road networks, in *Proc. of Int'l Symposium on Advances in Geographic Information Systems (GIS)*, pp. 1–8 (2003).

[33] Ji, X., Zha, H., Metzner, J. J., and Kesidis, G.: Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks, in *Proc. of Int'l Conf. on Communications (ICC)*, Vol. 7, pp. 3807–3811 (2004).

[34] Kalosha, H., Nayak, A., Ruhrup, S., and Stojmenovic, I.: Select-and-protest-based beaconless georouting with guaranteed delivery in wireless sensor networks, in *Proc. of Int'l Conf. on Computer Communications (INFOCOM)* (2008).

[35] Karp, B., and Kung, H.-T.: GPSR: Greedy perimeter stateless routing for wireless networks, in *Proc. of Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pp. 243–254 (2000).

[36] Ko, Y. B., and Vaidya, N. H.: Flooding-based geocasting protocols for mobile ad hoc networks, *Mobile Networks and Applications*, Vol. 7, No. 6, pp. 471–480 (2002).

[37] Komai, Y., Hara, T., and Nishio, S.: A Convex Hull Query Processing Method in MANETs, in *Proc. of Int'l Symposium on Reliable Distributed Systems (SRDS)*, pp. 331–332 (2014).

[38] Komai, Y., Hara, T., and Nishio, S.: Convex hull query processing methods in MANETs (in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 191–198 (2014).

[39] Komai, Y., Hara, T., and Nishio, S.: Processing convex hull queries in MANETs, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 64–73 (2015).

[40] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: A kNN query processing method in mobile ad hoc networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 287–288 (2011).

[41] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Searching k-nearest neighbor nodes in ad hoc networks (in Japanese), in *Proc. of DEIM Forum* (2011).

[42] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k nearest neighbor queries for location-dependent data in ad hoc networks (in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 17–25 (2012).

[43] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k nearest neighbor queries for duplicated location-dependent data in ad hoc networks (in Japanese), in *Proc. of IPSJ DICOMO Symposium*, pp. 553–560 (2013).

[44] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Processing k nearest neighbor queries for location-dependent data in MANETs, in *Proc. of Int'l Conf. on Database and Expert Systems Applications (DEXA), Part II*, pp. 213–227 (2013).

[45] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: k-nearest neighbor search based on node density in MANETs, *Mobile Information Systems*, Vol. 10, No. 4, pp. 385–405 (2014).

[46] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: KNN query processing methods in mobile ad hoc networks, *IEEE Trans. on Mobile Computing*, Vol. 13, No. 5, pp. 1090–1103 (2014).

[47] Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: k nearest neighbor search for location-dependent sensor data in MANETs, *IEEE Access*, Vol. 3, No. 1, pp. 942–954 (2015).

[48] Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A.: Geometric ad-hoc routing: Of theory and practice, in *Proc. of Int'l Symposium on Principles of Distributed Computing (PODC)*, pp. 63–72 (2003).

[49] Kuhn, F., Wattenhofer, R., and Zollinger, A.: Worst-case optimal and average-case efficient geometric ad-hoc routing, in *Proc. of Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 267–278 (2003).

[50] Lee, W.-C., and Zheng, B.: DSI: A fully distributed spatial index for location-based wireless broadcast services, in *Proc. of Int'l Conf. on Distributed Computing Systems (ICDCS)*, pp. 349–358 (2005).

[51] Li, J., Jannotti, J., De Couto, D. S., Karger, D. R., and Morris, R.: A scalable location service for geographic ad hoc routing, in *Proc. of Int'l Conf. on Mobile computing and networking (MobiCom)*, pp. 120–130ACM (2000).

[52] Liu, B., Lee, W.-C., and Lee, D. L.: Distributed caching of multi-dimensional data in mobile environments, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 229–233ACM (2005).

[53] Liu, H., Wan, P.-J., Jia, X., Liu, X., and Yao, F. F.: Efficient flooding scheme based on 1-hop information in mobile ad hoc networks, in *Proc. of Int'l Conf. on Computer Communications (INFOCOM)* (2006).

[54] Liu, X., Jia, X., Liu, H., and Feng, L.: A location aided flooding protocol for wireless ad hoc networks, in *Mobile Ad-Hoc and Sensor Networks*, pp. 302–313, Springer (2007).

[55] Manning, C. D., Raghavan, P., and Schütze, H. *et al.*: *Introduction to information retrieval*, Vol. 1, Cambridge university press Cambridge (2008).

[56] Matsuo, K., Goto, K., Kanzaki, A., Hara, T., and Nishio, S.: Overhearing-based efficient boundary detection in dense mobile wireless sensor networks, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 225–234 (2014).

[57] Nghiem, T. P., Waluyo, A. B., and Taniar, D.: A pure peer-to-peer approach for kNN query processing in mobile ad hoc networks, *Personal and Ubiquitous Computing*, Vol. 17, No. 5, pp. 973–985 (2013).

[58] Padmanabhan, P., Gruenwald, L., Vallur, A., and Atiquzzaman, M.: A survey of data replication techniques for mobile ad hoc network databases, *The VLDB Journal*, Vol. 17, No. 5, pp. 1143–1164 (2008).

[59] Perkins, C. E.: *Ad hoc networking*, Addison-Wesley Professional (2008).

[60] Ramany, V., and Bertok, P.: Replication of location-dependent data in mobile ad hoc networks, in *Proc. of Int'l Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, pp. 39–46 (2008).

[61] Roussopoulos, N., Kelley, S., and Vincent, F.: Nearest neighbor queries, in *Proc. of ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD)*, Vol. 24, pp. 71–79 (1995).

[62] Royer, E. M., and Toh, C.-K.: A review of current routing protocols for ad hoc mobile wireless networks, *IEEE Personal Communications*, Vol. 6, No. 2, pp. 46–55 (1999).

[63] Safar, M.: K nearest neighbor search in navigation systems, *Mobile Information Systems*, Vol. 1, No. 3, pp. 207–224 (2005).

[64] Sasaki, Y., Hagihara, R., Hara, T., Shinohara, M., and Nishio, S.: A top-k query method by estimating score distribution in mobile ad hoc networks, in *Proc. of Int'l Conf. on Advanced Information Networking and Applications (AINA) Workshops*, pp. 944–949 (2010).

[65] Sasaki, Y., Lee, W.-C., Hara, T., and Nishio, S.: An efficient index structure for location-based top-k search (in Japanese), in *Proc. of IPSJ DPS Workshop*, pp. 220–227 (2013).

[66] Scalable Network Technologies: Creators of QualNet Network Simulator Software: *http://web.scalable-networks.com/content/qualnet*.

[67] Song, L., and Wang, Y.: Multiple target counting and tracking using binary proximity sensors: bounds, coloring, and filter, in *Proc. of Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 397–406 (2014).

[68] Song, Z., and Roussopoulos, N.: K-nearest neighbor search for moving query point, in *Proc. of Int'l Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pp. 79–96, Springer (2001).

[69] The Official Bluetooth Wireless Info Site: *http://www.bluetooth.com/*.

[70] Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., and Sheu, J.-P.: The broadcast storm problem in a mobile ad hoc network, *Wireless Networks*, Vol. 8, No. 2-3, pp. 153–167 (2002).

[71] Tsuchida, G., and Ishihara, S.: Replica arrangement for location dependent data in consideration of network partition in ad hoc networks, *Int'l Journal of Communication Networks and Distributed Systems*, Vol. 2, No. 4, pp. 401–423 (2009).

[72] Tsuda, T., Komai, Y., Sasaki, Y., Hara, T., and Nishio, S.: Top-k query processing and malicious node identification against data replacement attack in MANETs, in *Proc. of Int'l Conf. on Mobile Data Management (MDM)*, pp. 279–288 (2014).

[73] Wi-Fi Peer-to-Peer: *http://developer.android.com/guide/topics/connectivity/wifip2p.html*.

[74] Wu, S.-H., Chuang, K.-T., Chen, C.-M., and Chen, M.-S.: Toward the optimal itinerary-based KNN query processing in mobile sensor networks, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 20, No. 12, pp. 1655–1668 (2008).

[75] Xia, S., Wu, H., and Jin, M.: Trace-routing in 3D wireless sensor networks: a deterministic approach with constant overhead, in *Proc. of Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 357–366 (2014).

[76] Xu, B., Vafaee, F., and Wolfson, O.: In-network query processing in mobile P2P databases, in *Proc. of Int'l Symposium on Advances in Geographic Information Systems (GIS)*, pp. 207–216 (2009).

[77] Xu, W., Chow, C.-Y., Yiu, M. L., Li, Q., and Poon, C. K.: MobiFeed: A location-aware news feed system for mobile users, in *Proc. of Int'l Symposium on Advances in Geographic Information Systems (GIS)*, pp. 538–541 (2012).

[78] Xu, Y., Fu, T.-Y., Lee, W.-C., and Winter, J.: Processing k nearest neighbor queries in location-aware sensor networks, *Signal Processing*, Vol. 87, No. 12, pp. 2861–2881 (2007).

[79] Xu, Y., Lee, W.-C., Xu, J., and Mitchell, G.: Processing window queries in wireless sensor networks, in *Proc. of Int'l Conf. on Data Engineering (ICDE)*, pp. 70–70 (2006).

[80] Yao, Y., Tang, X., and Lim, E.-P.: Continuous monitoring of kNN queries in wireless sensor networks, in *Proc. of Int'l Conf. on Mobile Ad-hoc and Sensor Networks (MSN)*, pp. 662–673, Springer (2006).

[81] Yao, Y., Tang, X., and Lim, E.-P.: Localized monitoring of kNN queries in wireless sensor networks, *The VLDB Journal*, Vol. 18, No. 1, pp. 99–117 (2009).

[82] Yashiro, T., and LaPorta, T.: Nomadic agent: Infrastructureless location based service system, *IPSJ Journal*, Vol. 46, No. 12, pp. 2952–2962 (2005).

[83] Yick, J., Mukherjee, B., and Ghosal, D.: Wireless sensor network survey, *Computer networks*, Vol. 52, No. 12, pp. 2292–2330 (2008).

[84] Yu, X., Pu, K. Q., and Koudas, N.: Monitoring k-nearest neighbor queries over moving objects, in *Proc. of Int'l Conf. on Data Engineering (ICDE)*, pp. 631–642 (2005).

[85] Zhou, H., Wu, H., and Jin, M.: A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks, in *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pp. 1602–1610 (2012).

[86] Zhou, H., Xia, S., Jin, M., and Wu, H.: Localized algorithm for precise boundary detection in 3D wireless networks, in *Proc. of Int'l Conf. on Distributed Computing Systems (ICDCS)*, pp. 744–753 (2010).

[87] Zhu, Q., Lee, D. L., and Lee, W.-C.: Collaborative caching for spatial queries in mobile P2P networks, in *Proc. of Int'l Conf. on Data Engineering (ICDE)*, pp. 279–290 (2011).

[88] Zhu, X., Sarkar, R., Gao, J., and Mitchell, J. S.: Light-weight contour tracking in wireless sensor networks, in *Proc. of Int'l Conf. on Computer Communications (INFOCOM)*, pp. 1849–1857 (2008).

[89] ZigBee Alliance: *http://www.zigbee.org/*.