A Robust Routing Method for Top-k Queries in Mobile Ad Hoc Networks

Daichi Amagata, Yuya Sasaki, Takahiro Hara, Shojiro Nishio
Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University
{amagata.daichi, sasaki.yuya, hara, nishio}@ist.osaka-u.ac.jp

***Abstract*—In mobile ad hoc networks (MANETs), to acquire only necessary data items, it is effective for each mobile node to retrieve data items using a top-k query, in which data items are ordered according to the score of particular attributes and the query-issuing mobile node acquires those data items with k-highest scores. In our previous work, we proposed a routing method for top-k query processing to reduce traffic while keeping highly accurate query result. In this method, each node holds a routing table consisting of the ranking of scores of data items in the network and identifiers of nodes to which queries are forwarded (*query addresses*) to obtain the necessary data items. When each mobile node issues a query, it refers to its own routing table, and unicasts the query message with requiring ranks of data items to each query address in its routing table. However, in highly dynamic networks, the accuracy of the query result decreases because the query transmission route to each data item is single. To achieve a high accuracy of the query result in highly dynamic networks, we propose in this paper, a routing method for top-k query processing that is an extension of our previous method. In our newly proposed method, each node does not unicast a query message with requiring ranks of data items, but multicasts the query message to the query addresses in its routing table. This method achieves robust query routing, due to multipath transmission of query messages, so that prevents from decreasing of the accuracy of the query result in highly dynamic networks. The simulation results show that our proposed method outperforms our previous method as to accuracy of the query result.**

***Keywords*-Top-k query; Routing; Mobile ad hoc networks;**

I. Introduction

Recently, mobile ad hoc networks (*MANETs*) [7], [11] have been attracting increasing interest. Query processing for retrieving only necessary data items is one of the key challenges in MANETs. Since the network bandwidth and batteries of mobile nodes in MANETs are limited, it is important for query processing to reduce data traffic by acquiring only the necessary data items. A possible and promising solution is for each mobile node to acquire data items using a top-k query, in which data items are ordered by the score of particular attributes and the query-issuing node acquires data items with the k-highest scores (*top-k result*).

A naive manner of processing a top-k query in a MANET is as follows. A query-issuing node floods all mobile nodes over the entire network with a query message. Each mobile node that receives the query message then transmits its own data items with the k-highest scores. Provided communication failures do not occur, the query-issuing node can in this way acquire all data items included in the top-k result. However, many data items that are not included in the top-k result are also sent back to the query-issuing node. This causes communication failures and excessive battery use.

In our previous works [5], [12], [13], we proposed query processing methods for top-k query to reduce traffic and also keep high accuracy of the query result. With these methods, the query-issuing node floods the query message attached with some score information. Each node then estimates the k-th score of the data item and sends back data items whose scores are equal to or greater than the estimated one. These methods can, therefore, decrease the number of transmitted data items. However, there is still a drawback that although the aim is to reduce traffic by not sending unnecessary data items, queries are forwarded to all nodes in the network by flooding. Thus, many nodes that do not contribute to the top-k result have to send redundant top-k queries and replies. As a result, there is a high level of traffic leading to excessive battery use and a risk of communication failures.

To solve this problem, we proposed a routing method for top-k query processing in MANETs [2] (we call this method the *previous method*). In the previous method, each node holds a routing table that comprises a ranking of scores of data items in the network and identifiers of nodes to which queries are forwarded (we call each of these nodes a *query address*) to obtain the necessary data items. When each mobile node issues a query, it first refers to its own routing table, and then sends the query message to the query addresses for this query by unicast. Nodes that receive the query message behave in the same way. By doing so, the previous method can acquire the top-k result with low traffic. However, if the network topology changes frequently, the accuracy of the query result decreases in the previous method because of the single query transmission path to each data item, which is too restrictive in MANETs.

To prevent from decreasing the accuracy of the query result in highly dynamic networks, in this paper we propose a routing method for top-k query processing in MANETs by extending our previous method. In the method we propose in this paper, the query-issuing node sends the query message to query addresses in its routing table by multicast. Each node that receives the query message determines the necessary query addresses that are worth transmitting by referring to its own routing table and the information included in the received query message, and then sends the query message to the necessary query addresses to obtain the top-k result. Unlike with the previous method, each node does not specify the requiring ranks of data items in the query message in the proposed method, so multiple paths to the nodes that have the top-k result are constructed in most cases. This gives our proposed method tolerance of highly network topology changes, while suppressing transmission the query messages to unnecessary query addresses. As a result, this method achieves robust query routing in highly dynamic networks. We have verified through simulations that our proposed method works very well in terms of the accuracy of the query result and reduced traffic.

The contributions of this paper are as follows.

- We propose an effective and robust routing method for top-k query processing to keep high accuracy of query result in highly dynamic networks.
- In our proposed method, each node holds a routing table to acquire the top-k result with low traffic. Even if link disconnections and updates of scores of data items occur, each node updates its own routing table accordingly. This procedure preserves the high accuracy of the query result.

The remainder of this paper is organized as follows: In Section II, we present some related works. In Section III, we describe the assumption. In Section IV, we present our previous method. In Section V, we present our proposed routing method for top-k query processing in MANETs. In Section VI, we show the results of our simulation experiments. Finally, in Section VII, we conclude our paper.

II. Related Work

In this section, we review existing research works on data-centric routing protocols and top-k query processing. We first survey some typical data-centric routing protocols. Since a data-centric routing protocol aims at retrieving data items with user-specified attribute, the approach of acquisition of data items could be similar to that in routing of top-k queries.

In [6], the authors have proposed a data-centric routing protocol called *directed diffusion* for wireless sensor networks. In this method, sinks disseminate a message stating their interest in data items. If nodes hold the data items with the interest, they forward the data packets toward the sinks by flooding. Through this process, efficient paths between sinks and nodes are reinforced by a reinforcement mechanism. Messages and data packets are transmitted through the reinforced paths with high probabilities. This method can construct efficient paths and achieve low traffic for data transmission. However, this method is not effective in MANETs, in which the network topology changes dynamically. In [9], the authors proposed a routing method for top-k query processing in wireless sensor networks. This method constructs a cluster for top-k queries whose size is determined based on the residual energy of the cluster head. This method can maintain a balance between energy consumption among sensor nodes in the network by changing the cluster head. However, the cost of cluster maintenance is excessive in MANETs due to the movement of nodes.

A large number of strategies for processing a top-k query have been proposed in several research fields, such as unstructured P2P networks [1] and wireless sensor networks [8], [14], [15]. Since top-k query processing in neither network considers the movement of nodes, they cannot directly adapt to MANETs. However, there is no research that focuses on top-k query processing in MANETs except for our previous work and EcoTop [10]. Since the assumption in [10] is quite different to the assumption in this paper, we describe here our previous work, 2-phase top-k query processing [13], which can keep the highest accuracy of the query result of our previous methods.

2-phase top-k query processing consists of two phases: the *search* and *data collection* phases. In the search phase, the query-issuing node floods a query message, and then, collects the information on the scores of data items held by all nodes over the entire MANET (the size of any score is smaller than that of the data items). It then checks those scores and determines the k-th score as the threshold. In the data collection phase, the query-issuing node floods a query message attached with the threshold, and then, collects data items with scores equal to or larger than the threshold. This method can decrease the number of transmitted data items as much as possible. However, all nodes receive and forward queries (and replies) twice. This increases unnecessary traffic. To solve this problem, in our previous method [2] and the method proposed in this paper, each node forwards a query message based on its own routing table to only nodes that contribute to the top-k result. As a result, our method can decrease traffic and also keep high accuracy of the query result. We describe the previous method in detail in Section IV.

III. Assumption

The system environment is assumed to be a MANET in which mobile nodes retrieve data items held by itself and other mobile nodes using a top-k query. The query-issuing node transmits a query message with the query condition and the number of requested data items, k, and acquires data items with the k-highest scores among all data items held by mobile nodes over the entire network. Each mobile node determines k, from 1 to k_{max}.

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 $\boldsymbol{D}= \{D_1, D_2, \cdots, D_n\}$, where n is the total number of data items and D_i $(1 \leq i \leq n)$ is a data identifier. Each data item is held by a specific node, and scores of data items are updated as time passes. For simplicity, all the data items are of the same size and not replicated. The scores of data items can be calculated from the query condition and certain scoring functions.

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

Table I
SCORES OF DATA ITEMS HELD BY EACH MOBILE NODE

Node identifier	Scores of data items
M_1	89, 88, 82, 78, 70, 66, 49, 30, 21
M_2	99, 81, 80, 77, 72, 71, 69, 55, 42
M_3	90, 79, 53, 41, 33, 29, 19, 12, 11
M_4	95, 87, 76, 61, 58, 50, 44, 37, 10
M_5	85, 83, 74, 67, 56, 23, 18, 15, 13
M_6	68, 62, 60, 59, 56, 40, 30, 25, 22

Table II
M_1'S ROUTING TABLE

Rank	Score of data item	Data item's holder	Query address	Time stamp
1	99	M_2	M_2	10:00
2	95	M_4	M_2	10:00
3	90	M_3	M_3	10:00
⋮	⋮	⋮	⋮	⋮
20	69	M_2	M_2	10:00

IV. PREVIOUS METHOD

Our proposed method in this paper is an extension of our previous method [2]. Therefore, before explaining the proposed method, in this section we explain our previous method and describe problems with it.

A. Overview

In our previous method, a top-k query is processed based on a routing table. Hence, it is first necessary to construct a routing table at each node.

The routing table is constructed by collecting the information on the scores of data items, and consists of the ranking of scores of data items (*rank table*), the identifiers of nodes that hold data items with high scores (*data holders*), the identifiers of nodes to which a query is forwarded to acquire data items with high scores (*query addresses*), and time stamps (we describe this procedure in detail in Section IV-B). In top-k query processing, the query-issuing node sends a query message to the query addresses attached with ranks of the data items that it requires. If the nodes that receive the message hold the required data items, they reply with the data items. If a link disconnection occurs while processing a top-k query, the node that detects the disconnection searches for another route and updates its own routing table accordingly. When the score of a data item is updated, the node that updated the score transmits a message to the nodes that hold data items whose ranks have changed (we describe this procedure in detail in Section IV-D).

B. Routing Table Construction

In this section, we explain in detail how the routing table is constructed. If the query-issuing node does not have its own routing table that corresponds to the query condition, it starts constructing one. The procedure is like a naive manner introduced in Section I. Each node sends back a message with the information on their data items (the score and the node identifier of the holder) with $N(> k_{max})$-highest scores to its parent. N is set at larger than k_{max} to be able to cope with a case in which the scores of data items with k_{max} highest scores are updated to smaller scores than the k_{max}-th score. In addition, each node that receives the message stores the received information on data items with the N-highest scores and the sender node of the message, so it knows which data item is replied from which node (we call this information the *node-data list*). From this procedure, the query-issuing node can make a *routing table* that consists of the scores of data items with N-highest scores (rank table), node identifiers of data holders, node identifiers of query addresses, and time stamps. The time stamp is set as the time when the routing table is constructed. The list of this information in the routing table is sorted in descending order by score. Intermediate nodes can also make a node-data list for constructing the routing table.

The query-issuing node then floods a message attached with the rank table, data holders, and time stamps. Nodes that receive the message for the first time construct their own routing table. The process by which a receiver of this message constructs the routing table is as follows.

1) The receiver stores the information in the received rank table, data holders, and time stamps in its own routing table.
2) The receiver sets query addresses for all ranks in its own routing table as the identifier of the node that sent the rank table to itself.
3) The receiver updates query addresses for ranks whose data holder is itself to its own identifier.
4) The receiver further updates query addresses for ranks to the identifiers of nodes corresponding to data items in its node-data list.

After the above procedure, node that holds data items with k_{max}-highest scores floods a message attached with its node identifier to construct the shortest path to the data items. This process is important to reduce traffic in top-k query processing, since the routing table depends on the position of the root node without this procedure (i.e., the query-issuing node).

As a result of all the above procedure, each node knows which one is the next hop node for retrieving the data items with the required ranks via the shortest path. Tables I and II show respectively the scores of data items held by mobile nodes and the routing table held by mobile node, M_1, when N=20 (the network topology is shown in Figure 1). For example, M_1 knows that the next hop node is M_2 to retrieve the first rank data item.

C. Top-k Query Processing

In our previous method, each node unicasts a query message which specifies the requiring ranks of data items to query addresses that corresponds to the ranks in its routing table. Each node that receives this message replies only with the required data items. Thus, this method can prevent the transmission of query messages to nodes that do not contribute to the top-k result and the sending back of unnecessary data items. In this method, when nodes receive reply messages from the query addresses in their routing table, they update the time stamps of the corresponding query addresses to the current time. Moreover, nodes that receive the reply message refer to

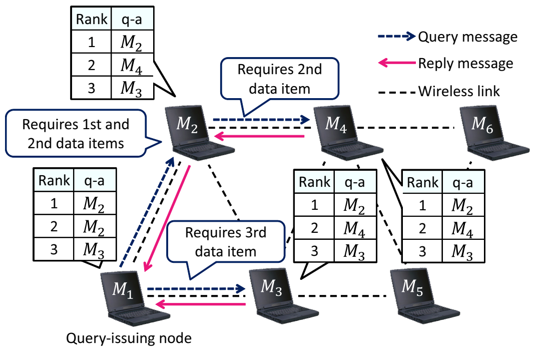


Figure 1. Example of top-k query processing in the previous method

the node identifier of the reply sender and its routing table, and if the node identifier of the reply sender is different from the query address corresponding to the rank of the received data item, the nodes update the query address to the sender.

Figure 1 shows an example of top-k query processing in the previous method when M_1 issues a top-k query for acquiring data items with the three highest scores ($k = 3$). In the Figure, the table in each balloon denotes a part of routing table held by each node, and 'q-a' denotes the query address. M_1 sends a query message to M_2 to request the 1st and 2nd rank data items and M_3 to request the 3rd rank data item. Then, for example, since M_2 knows that the 1st data item's holder is itself, and the 2nd one is not, it transmits a query message to request the 2nd data item to M_4. Nodes that hold required data items send them back. Finally, M_1 can acquire the data items with the three highest scores.

D. Routing Table Maintenance

In our previous method, to maintain the accuracy of the routing table, each node has to update its own routing table if a link disconnection occurs and the scores of data items are updated. In this section, we explain how a routing table is maintained.

1) Handling Link Disconnection: In MANETs, the network topology dynamically changes due to movement of nodes. If a mobile node cannot transmit a query to a query address or a reply to its parent because of a link disconnection between these nodes, the accuracy of the query result decreases. Therefore, a mobile node that has detected the link disconnection (we call this node the *route request node*) searches for another route to the query address or its parent (we call the query address and parent the *destination node* in this section) by sending a *route request message*.

The procedure for searching for the destination node in both query and reply transmission is similar. The route request node broadcasts a *route request message* with a TTL that represents a range to search the destination node by hop count. If a node that receives this message is the destination node or the query-issuing node (only for reply transmission), it starts sending back a *route reply message*. Otherwise, it decrements the TTL, and rebroadcasts the route request message if the TTL is greater than 0.

In this way, even if a link disconnection occurs, mobile nodes can continue to transmit query and reply messages by finding another route, and thus the accuracy of the query result can be kept.

2) Score Update Announcement: If scores of data items are updated, the ranking of data items may change. In our privious method, when the ranking of data items changes, the mobile node that updated the data items updates its own routing table, and sends a message to notify all nodes of the change of score. There are three cases in which the routing table needs to be maintained when updating the score of a data item D_u held by a mobile node. The algorithm for maintaining the routing table is shown in Algorithm 1.

Algorithm 1 Routing Table Update Algorithm

```
1: if $D_u$ moves to different rank in the routing table then
2:     Resort the recorded information in descending order by score
3: else if $D_u$ is newly included in the routing table then
4:     Insert the information on $D_u$
5:     Move down the rank of each data item whose score is smaller $D_u$
6: else if $D_u$ is excluded from the routing table then
7:     Delete the information on $D_u$
8:     Raise the rank of each data item whose score is smaller than $D_u$
9: end if
```

The node then (in all the cases) sends the *score update announcement message* to query addresses corresponding to data items whose ranks have changed. Each node that receives a score update announcement message updates its own routing table according to Algorithm 1. Here, it sets query addresses for changed ranks to the node identifier of the sender of the score update announcement message. Next, it forwards the score update announcement message the same way as the node which updated the score of data items. This message is transmitted until the data holder of the data item whose rank has changed receives the message.

In this way, a node that updates the score of a data item transmits a score update announcement message to only nodes that need to update their routing table as soon as possible (i.e., nodes holding data items whose ranks have changed). Nodes which do not receive a score update announcement message can update their own routing table when they receive a reply message in a future top-k query. Therefore, this procedure suppresses overhead for transmission of score update announcement messages, but keeps high accuracy of the query result.

E. Problems with the Previous Method

Our previous method can suppress unnecessary traffic by sending a query message to only nodes which contribute to the top-k result. However, this method cannot keep its performance if mobile nodes move fast (i.e., in highly dynamic networks). Our previous method cannot adapt to highly dynamic networks for the following reasons.

- Unicasting a query message
  In our previous method, each node unicasts a query message that specifies the requiring ranks of data items to each query address. The query transmission route to each of data items in the top-k result is a single path. However, since nodes move freely in MANETs, the network topology dynamically changes, and thus, query transmission along a single path cannot be counted upon to provide reliable transmission.
- Inefficient route request processing
  In our previous method, if nodes cannot forward a query message to the query address due to a link disconnection, they search for another route to the query address by means of a route request. This procedure needs a round-trip message transmission, which causes a delay. As a result, another link disconnection may occur due to this delay.

If the network topology changes frequently, it is difficult to accurately reflect the latest network topology to a routing table. A more robust routing method is therefore needed.

V. Proposed Method

In this section, we describe our proposed method. First, we describe our ideas for extending our previous method. We then explain how to process a top-k query and how to maintain the routing table in our proposed method.

A. Ideas for Extension

Our proposed method utilizes the same routing table as our previous method, but a time stamp is not included. In our previous method, each node specifies the requiring ranks of data items to narrow down the query transmission route to each rank to be single. However, if the query transmission route is single, the query delivery ratio falls if the network topology changes frequently. This makes it more effective to transmit a query message by multipath to improve the query delivery ratio. If each node unicasts the query message to all the query addresses included within the k-th rank in its routing table, the query message is transmitted by multipath. However, if the number of query addresses is large, sending the query message by unicast increases the traffic and the delay. Hence, in our proposed method, each node sends a query message to all query addresses included within the k-th rank in its routing table at the same time by multicast. Moreover, it determines unnecessary query addresses to transmit the query message by referring to its own routing table and the information included in the received and overheard query messages. This achieves robust transmission of query message by multipath but suppresses redundant query transmission.

Here, constructing the shortest path described in Section IV-B is effective in our previous method, because the query message is sent to each rank of data item via a single path. However, redundant routes are produced in our proposed method, and thus, constructing the shortest path has less meaning. Therefore, our proposed method does not construct the shortest path.

B. Top-k Query Processing

In our proposed method, each node sends a query message based on its own routing table and the information in the received messages. Nodes that receive the query message send back only data items with the k-highest scores in their routing table. We explain below the details of the top-k query processing.

1) Transmission of query: A query message consists of the following elements.

- *Query-issuing node's ID*: the node identifier of the query-issuing node.
- *Query ID*: the identifier of the query.
- *Number of requested data items*: the number of data items, k, that the query-issuing node requests.
- *Query condition*: the query condition specified by the query-issuing node.
- *Query path*: the list of identifiers of mobile nodes on the path through which the query message is transmitted.
- *Query address list (QL_s)*: the list of identifiers of mobile nodes to which the sender node transmits this message.
- *Parent's query address list (QL_p)*: the query address list which is included in the query message received from the parent.

First of all, the query-issuing node, M_p, specifies the query condition and checks whether it has the routing table or not. If not, it performs the procedure in Section IV-B. Otherwise or after constructing the routing table, M_p specifies the number of requested data items, k, and sets all query addresses included within the k-th ranks in its routing table as QL_s. It then broadcasts a query message. In this message, QL_p is set as NULL, the query-issuing node's ID is set as M_p, and the query path is set as M_p.

The node, M_q, that receives the query message performs according to Algorithm 2. In Algorithm 2, RoutingTable.QueryAddress[i] denotes that i-th query address included in M_q's routing table. NL_{query} is the list of pairs of the identifier of the sender node and its hop count from the query-issuing node. When receiving the query message for the first time, each node sends an ACK to its parent, and nodes received it set its sender node as their child. Thus, each node can know which node sends back a reply message to it.

All nodes that received the query message try to acquire data items with k-highest scores (line 4 in Algorithm 2), so this method can transmit the query message by multipath. In addition, each node that received the query message can know which nodes have already been transmitted the query message from the received QL_s, QL_p, and query path. Therefore, when it sends the query message, it includes these nodes as QL_s in its query message, which can suppress redundant query transmission (line 4-8 in Algorithm 2). Thus, the query delivery ratio is improved

Algorithm 2 Query Transmission Algorithm

```
1:  /* On receiving the query*/
2:  if $M_q$ receives the query for the first time then
3:      Set the sender node as parent
4:      for $i = 0$ to $k$ do
5:          if RoutingTable.QueryAddress[i] is not included
            in the received $QL_s, QL_p$, and query path then
6:              Add $i$-th query address to $M_q$'s $QL_s$
7:          end if
8:      end for
9:      if $M_q$'s $QL_s$ =NULL then
10:         Send reply message to the parent
11:     else
12:         Set query timer
13:     end if
14: else
15:     Add the sender node to $NL_{query}$
16:     if the sender node is included in $M_q$'s $QL_s$ then
17:         Delete it from $M_q$'s $QL_s$
18:     end if
19: end if
20: /* On sending a query*/
21: if query timer expired then
22:     if $M_q$'s $QL_s$! =NULL then
23:         Set the received $QL_s$ to $M_q$'s $QL_p$
24:         Add $M_q$'s identifier to query path
25:         Send query message
26:     else
27:         Send reply message to the parent
28:     end if
29: end if
```

over our previous method while suppressing unnecessary transmission, which achieves robust query routing.

2) Transmission of reply: In our proposed method, each mobile node sends back a reply message attached with data items with k-highest scores in its routing table, which can suppress sending back unnecessary data items which are not included in the top-k result.

A reply message consists of the following elements.

- *Query ID*: the identifier of the query.
- *Sender node's ID*: the node identifier of the node that sends the reply message.
- *Data list*: the list of pairs of data items and their scores.

Each mobile node, M_r, whose QL_s is NULL in its query message starts sending a reply message to its parent. The procedure of reply transmission is presented in Algorithm 3.

Through this procedure, each mobile node sends back only data items that are included in the top-k result, so unnecessary data items are not transmitted. Moreover, each mobile node that received replies can update their routing table (line 2-10 in Algorithm 3), which can keep accuracy of the routing table. In addition, if mobile nodes detect a link disconnection with their parent, they can transmit the reply message to another neighboring node by choosing one whose hop count is the minimum in its NL_{query}.

Algorithm 3 Reply Transmission Algorithm

```
1:  /* On receiving a reply*/
2:  Compare query address (corresponding to the received
    date items) to the sender node
3:  if query address != sender node then
4:      if query address = $M_r$'s child then
            Delete the query address from $M_r$'s children
5:      end if
6:      query address ← sender node
7:  end if
8:  if $M_r$ receives a data item which is not recorded in
    $M_r$'s routing table then
9:      Update its own routing table (as described in Sec-
        tion IV-D2)
10: end if
11: /* On sending a reply*/
12: if $M_r$ receives a reply message from all children ||
    pre-determined time limit has elapsed then
13:     Add all received data items and $M_r$'s data items
        whose scores are within the $k$-th rank to $M_r$'s reply
        message
14:     Send reply message to its parent
15: end if
```

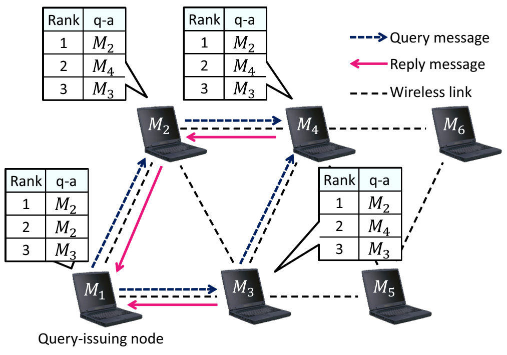

Rank q-a 1 M_2 2 M_4 3 M_3
Rank q-a 1 M_2 2 M_4 3 M_3
Query message
Reply message
Wireless link
M_2
M_4
M_6
Rank q-a 1 M_2 2 M_2 3 M_3
Rank q-a 1 M_2 2 M_4 3 M_3
M_1
M_3
M_5
Query-issuing node

Figure 2. Example of a behavior of the proposed method

This is different from the previous method. Since this minimizes any reply transmission delay caused by finding another route to its parent or the query-issuing node, it gives our proposed method tolerance to the network topology change.

Figure 2 shows an example of the behavior of the proposed method when M_1 issues a top-k query for acquiring data items with the three highest scores (k=3) in the same situation as in Figure 1. M_1 refers to its own routing table and sends a query message to M_2 and M_3 because they are included within k-th ranks in M_1's routing table. M_2 and M_3 behave the same way as M_1, so M_2's QL_s consists of M_3 and M_4, and M_3's QL_s consists of M_2 and M_4. Here, M_2 removes M_3 from its QL_s and M_3 removes M_2 from its QL_s because the received QL_s consists of M_2 and M_3. They then send a query message to M_4. In this case, we assume that M_4 receives the query message from M_2 for the first time, it adds M_3 to NL_{query}. M_3 and M_4 start sending a reply message to their parent (M_1 and M_2 respectively) because they have no query address. Through this procedure, M_1 can acquire the top-k result.

C. Routing Table Maintenance

In this section, we explain how to maintain the routing table in our proposed method.

1) Handling Link Disconnection: Firstly, we explain how to update a routing table if a node detects a link disconnection with query addresses.

If a node that detects a link disconnection with query addresses during query transmission (we call this node a *route request node*), it searches for another route to them (we call these nodes *destination nodes*) by sending a route request message ($RREQ$). In our proposed method, information on the top-k query message is attached to the RREQ, which can suppress a query transmission delay as possible because it does not need to send a top-k query message after it received a reply of RREQ ($RREP$). The route request algorithm for node M_s is presented in Algorithm 4. Basically, the procedure of route request is same as our previous method. However, our proposed method can transmit the query message by multipath, so even if nodes that receive an RREQ are not the destination node, they can send back the RREP if they know the destination node as their children and neighboring node (NL_{query}). That is, they know the destination node that has already received the query message (line 9-10 in Algorithm 4), which reduces traffic and delays by suppressing unnecessary RREQ transmission. Moreover, unlike with our previous method, an RREQ has the information on the top-k query, so it corresponds to transmit the RREQ and the query message at the same time, which suppresses query transmission delay.

2) Score Update Announcement: In our previous method, the node that updates the scores of data items sends a score update announcement message to query addresses corresponding to data items whose ranks have changed. However, it makes a roundabout route to the data holders because each query message to nodes that hold newly ranked data items is once forwarded to nodes that have data items which have been updated and caused the change of the ranks. This increases unnecessary traffic. Therefore, in our proposed method, the node that updates the scores of data items sends a score update announcement message to those query addresses that are included within k_{max}-th ranks in its own routing table. Each node that receives this message updates its own routing table accordingly (described in Section IV-D2), and transmits this message in the same way as query transmission. Moreover, it prevents from constructing a roundabout route by overhearing the score update announcement message and each node updates the query address if the sender of the score update announcement is the data holder.

Algorithm 4 Route Request Algorithm

```
1:  /* On receiving RREQ*/
2:  if $M_s$ receives RREQ for the first time then
3:      Store the sender node
4:      if destination node is itself then
5:          Send RREP to the stored sender node of the RREQ
6:          if $M_s$ has not received the query message then
7:              Perform the procedure of Algorithm 2
8:          end if
9:      else if destination node is included in $NL_{query}$ or $M_s$'s children then
10:         Send RREP to the stored sender node of the RREQ
11:     else if RREQ.TTL $>$ 0 then
12:         Decrement TTL
13:         Broadcast RREQ
14:     end if
15: end if
16: /* On receiving RREP*/
17: Update the query addresses of corresponding (destination node's) ranks to the sender node
18: if $M_s$ is the route request node then
19:     if all destination nodes have already received the query message then
20:         Send reply message to its parent
21:     end if
22: else
23:     Send RREP to the stored sender node of the RREQ
24: end if
```

VI. Simulation Experiments

In this section, we show the results of simulation experiments regarding the performance evaluation of our proposed method. For the simulation experiments, we used a network simulator, QualNet4.0 [16].

A. Simulation Model

The number of mobile nodes in the entire system is 100 ($M_1, M_2, \cdots, M_{100}$). These mobile nodes exist in an area of 800 [m] × 500 [m] and move according to the random waypoint model [3] where the movement speed and the pause time are v [m/sec] and 60 [sec] respectively. Each mobile node transmits messages and data items using an IEEE 802.11b device whose data transmission rate is 11 [Mbps]. The transmission power of each mobile node is determined so that the radio communication range becomes about 100 [m]. Packet losses and delays occur due to a radio interference. Each mobile node has 20 data items, and the size of these data items is 128 [byte]. The score distribution in the entire network follows the normal distribution and the total range of scores are set as [1, 999]. A node that is randomly selected at every T [sec] updates the scores of data items held by itself. In our proposed method, the number of rank orders, N, maintained in the routing table and k_{max} are respectively set as 100 and 50. The TTL value of a route request message is set as 2.

Table III shows the parameters and their values used in the simulation experiments. The parameters are basically fixed at constant values specified by the numbers to the left of the parenthetic values. Each of them is varied in the range specified by the parenthetic element in a simulation experiment.

We compared the performance of our proposed method

Table III
CONFIGURATION OF PARAMETERS

Symbol	Meaning	Values(Range)
k	Number of requested data items	30 (1∼50)
v	Mobile velocity of nodes	0.5 (0∼3.0)[m/sec]
T	Score update interval	90 (15∼300)[sec]

with that of our previous method and the method proposed in [13] (denoted by 2-phase).

In the above simulation model, we randomly determined the initial position of each mobile node and randomly determined a query-issuing node at every 3 [sec]. We evaluated the following criteria over a period of 600 [sec].

- *Accuracy of query result:* is MAP (*Mean Average Precision*) which is often used to calculate the accuracy with a rank. MAP averages accuracy of each query result, AP (*Average Precision*). AP and MAP are determined by the following equations.

$$AP_i = \frac{1}{k}\sum_{j=1}^{k}\frac{x}{j}\cdot e \tag{1}$$

$$MAP = \frac{1}{querynum}\sum_{i=1}^{querynum} AP_i \tag{2}$$

  AP_i is the accuracy of the i-th query result. x is the number of collected data items with the j-highest scores included within the j-highest correct answer set. $querynum$ is the total number of issued queries. e is determined by the following equation.

$$e = \begin{cases} 1 & (j\text{-th answer is included in top-k result}) \\ 0 & (\text{otherwise}) \end{cases} \tag{3}$$

  Thus, MAP rises as the query-issuing node obtains data items with higher score.
- *Traffic:* is defined as the total volume of messages sent during the simulation.
- *Search time:* is defined as the average of the elapsed time between the query-issuing node issuing a top-k query and its acquisition of the result.

B. Impact of Number of Requested Data Items

We examined the effects of the number of requested data items, k. Figure 3 shows the simulation result. In the graphs, the horizontal axis indicates k. The vertical axes indicate, respectively, the accuracy of query result in Figure 3(a), the traffic in Figure 3(b), and the search time in Figure 3(c).

From Figure 3(a), we can see that our proposed method keeps high accuracy of query result compared with the other methods. This is because the query and reply delivery ratios are high due to the efficient query transmission by multipath and route request procedure. In 2-phase, there is the case that the threshold is not an accurate value because of packet losses in the search phase. Therefore unnecessary traffic increases due to the sending back of data items which are not included in the top-k result, which causes packet losses and communication failures.

From Figure 3(b), we can see that the traffic increases in all methods as k increases. This is because the number of data items that are attached to the reply message increases, and thus the size of the reply message becomes larger. In our previous method, each node unicasts a query message with respect to each query address, which increases the number of sending query messages. On the other hand, in our proposed method, each node sends a query message to multiple query addresses by multicast (i.e., only one packet). Therefore, while our proposed method produces multipath, it can suppress the traffic for sending query messages. Moreover, in our previous method, if nodes detect a link disconnection with their parent while transmitting the reply message, they perform the route request procedure. On the other hand, in our proposed method, if nodes know their neighboring nodes (i.e., NL_{query}), they transmit the reply message to a neighboring node without the route request procedure. Thus, the traffic of our proposed method is smaller than that of our previous method.

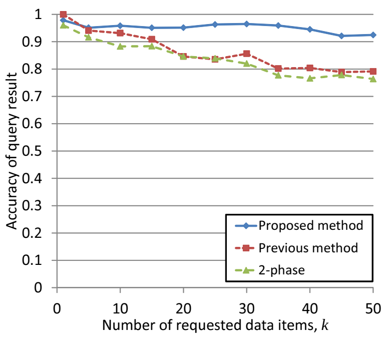
From Figure 3(c), we can see that the search time of our proposed method is the shortest of all the methods. In our proposed and previous methods, the number of each node's children is smaller than that in 2-phase, so the time that waits for the reply message from the children is shortened. On the other hand, the limited query transmission causes frequent route request procedures in our previous method. These are the reasons why the search time of our proposed method is the shortest.

C. Impact of Mobile Velocity of Nodes

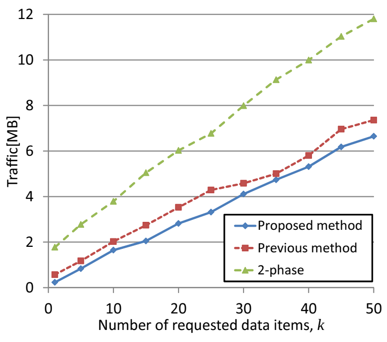
We examined the effects of the mobile velocity of nodes, v. Figure 4 shows the simulation result. In the graphs, the horizontal axis indicates v. The vertical axes indicate, respectively, the accuracy of query result in Figure 4(a), the traffic in Figure 4(b), and the search time in Figure 4(c).

From Figure 4(a), we can see that our proposed method keeps high accuracy of query result compared with our previous method. This is because the query transmission route to intermediate nodes and data holders become multipath in our proposed method, which improves the query delivery ratio. Additionally, when each node performs a route request procedure, our proposed method suppresses the query transmission delay as much as possible by attaching the information on top-k query to the RREQ, which has tolerance of the network topology change.

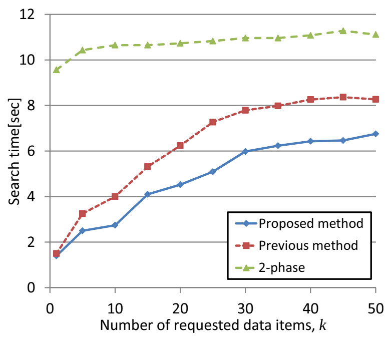
From Figure 4(b), we can see that the traffic increases in our proposed and previous methods as v increases. This is because as v increases, the link disconnections occur frequently, so many nodes perform the route request procedure in our proposed and previous methods. In 2-phase, query message is forwarded by flooding and there is the case that unnecessary data items are sent back. Therefore, the traffic of 2-phase is larger than that of the two methods. When v is large, the traffic of our proposed method is larger than that of our previous method. In our proposed method, the traffic for route request in query



(a) Accuracy of query result

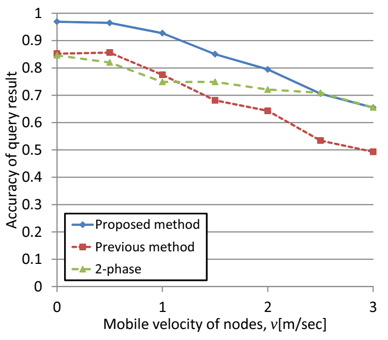


(b) Traffic

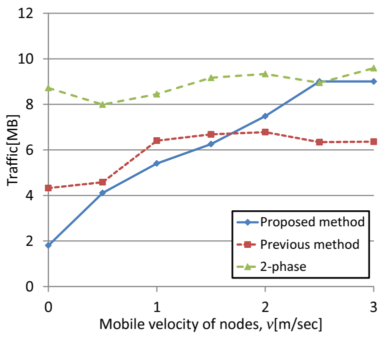


(c) Search Time

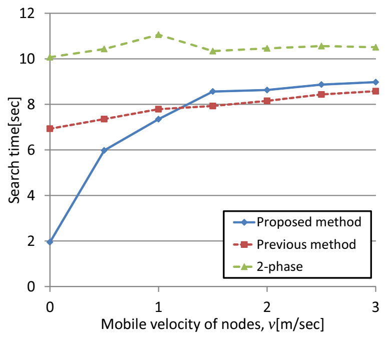
Figure 3. Effects of k



(a) Accuracy of query result



(b) Traffic



(c) Search Time

Figure 4. Effects of v

transmission is larger than in our previous method because an RREQ is attached with the information on top-k query. In addition, when a link disconnection with its parent occurs, a node that knows its neighboring nodes can send a reply message to one of them in our proposed method. Moreover, since query delivery ratio is low in our previous method, some data holders cannot receive query messages and send back reply messages. Thus, the number of sending reply messages in our proposed method is larger than that in our previous method.

From Figure 4(c), we can see that the search time becomes longer in our proposed and previous methods as v increases. The reason is that link disconnections frequently occur, and a risk that a reply message is dropped at intermediate nodes increases as v increases. Hence, many intermediate nodes wait for the reply message until the pre-determined time elapses. The search time in 2-phase is longer than our proposed and previous methods because it requires two rounds of message transmission.

D. *Impact of Score Update Interval*

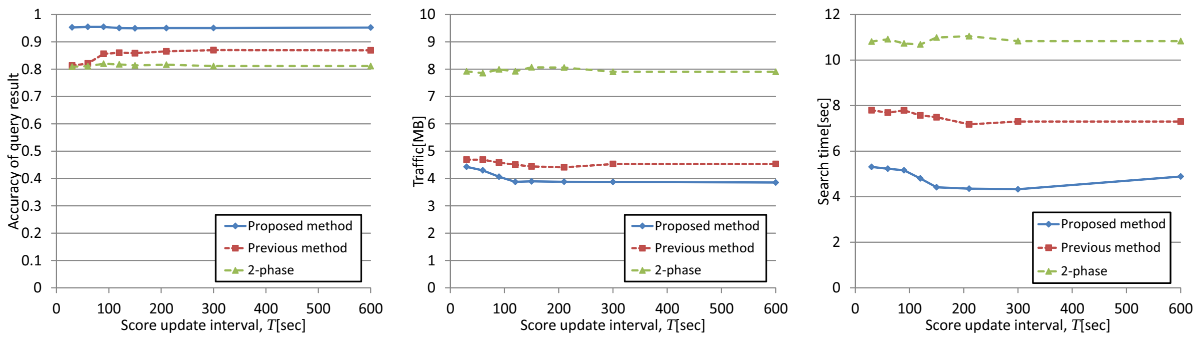
We examined the effects of the score update interval, T. Figure 5 shows the simulation result. In the graphs, the horizontal axis indicates T. The vertical axes indicate, respectively, the accuracy of query result in Figure 5(a), the traffic in Figure 5(b), and the search time in Figure 5(c).

From Figure 5(a), we can see that our proposed method keeps high accuracy of query result even if T is short. In our proposed method, a score update announcement is transmitted along the routes that top-k query messages are frequently transmitted. This is because the node which updates the score of a data item sends the score update announcement to query addresses which are included within the k_{max} ranks in its routing table. In other words, the query addresses are often updated in order to successfully transmit query messages. Therefore, the message delivery ratio is high and many nodes can update their routing table accurately.

From Figures 5(b) and 5(c), we can see that the traffic of our proposed method increases and search time lengthens when T is short, while these are smaller than the other methods. This is because a score update announcement is transmitted to query addresses within the k_{max} ranks in the routing table, and thus, there is a tendency for message collisions to occur between a query message and a score update announcement.

VII. Conclusion

In this paper, we proposed a routing method for top-k query processing in MANETs that is an extension of our previous method. In this method, each mobile node holds a routing table and knows the nodes to which queries are forwarded (query addresses). Each node sends a query message to the nodes which are included within k-th rank in its own routing table as query addresses. By doing so, multiple paths are constructed in most cases unlike our previous method, which improves the query delivery ratio. In addition, each node selects query addresses by referring to the received query messages, and then sends the query message to nodes that are worth transmitting it. Nodes that receive the query message send back only data items with k-highest scores in its own routing table. This eliminates unnecessary traffic by not sending data items that are not included in the top-k result. If a mobile



(a) Accuracy of query result (b) Traffic (c) Search Time

Figure 5. Effects of T

node detects a link disconnection during the transmission of query and reply messages, it finds another route and updates its own routing table. When the score of a data item is updated, a score update announcement is efficiently forwarded. The simulation results show that our proposed method outperforms our previous method in almost all cases.

Even in our proposed method, the probability of success of a route request becomes lower if the node density is small. To solve this problem, we currently plan to further extend our proposed method by replicating data items.

Acknowledgment

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

References

[1] R. Akbarinia, E. Pacitti, and P. Valduriez, “Reducing network traffic in unstructured P2P systems using top-k queries,” Distributed and Parallel Databases, vol.19, no.2–3, pp.67–86, 2006.
[2] D. Amagata, Y. Sasaki, T. Hara, and S. Nishio, “A routing method for top-k query processing in mobile ad hoc networks,” Proc. Int. Conf. on Advanced Information Networking and Applications, 2013 (to appear).
[3] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc networks research”, Wireless Communications and Mobile Computing: Special issue on Mobile Ad Hoc Networking: Reserach, Trends and Applications, vol.2, no.5, pp.483–502, 2002.
[4] J. Cecílio, J. Costa, and P. Furtado, “Survey on data routing in wireless sensor networks,” Wireless Sensor Network Technologies for the Information Explosion Era, vol.278, pp.3–46, 2010.
[5] R. Hagihara, M. Shinohara, T. Hara, and S. Nishio, “A message processing method for top-k query for traffic reduction in ad hoc networks,” Proc. Int. Conf. on MDM, pp.11–20, 2009.
[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion for wireless sensor networking,” IEEE/ACM Trans. Networking, vol.11, no.1, pp.2–16, 2003.
[7] D. B. Johnson, “Routing in ad hoc networks of Mobile Hosts,” Proc. IEEE WMCSA’94, pp.158–163, 1994.
[8] X. Liu, J. Xu, and W. -C. Lee, “A cross pruning framework for top-k data collection in wireless sensor network,” Proc. Int. Conf. on MDM, pp.157–166, 2010.
[9] S. Mo, H. Chen, and Y. Lie, “Clustering-based routing for top-k querying in wireless sensor networks,” EURASIP Journal on Wireless Communications and Networking, vol.2011, no.1, pp.1–13, 2011.
[10] N. Padhariya, A. Mondal, V. Goyal, R. Shankar, and S. K. Madria, “EcoTop: an economic model for dynamic processing of top-k queries in mobile-P2P networks,” Proc. Int. Conf. DASFAA, pp.251–265, 2011.
[11] C. E. Perkins, and E. M. Ooyer, “Ad-hoc on-demand distance vecotr routing,” Proc. iEEE WMCSA’99, pp.90–100, 1999.
[12] Y. Sasaki, R. Hagihara, T. Hara, and S. Nishio, “A top-k query method by estimating score distribution in mobile ad hoc networks,” Int. Workshop on Data Management for wireless amd Pervasive Communications, pp.944–949, 2010.
[13] Y. Sasaki, T. Hara, and S. Nishio, “Two-phase top-k query processing in mobile ad hoc networks,” Proc. Int. Conf. on Network-Based Information Systems, pp.42–49, 2011.
[14] M. Wu, J. Xu, X. Tang, and W. -C. Lee, “Top-k monitoring in wireless sensor networks,” IEEE Trans. Knowledge and Data Engineering, vol.19, no.7, pp.962–976, 2007
[15] M. Ye, X. Liu, W. -C. Lee, and D. -L. Lee, “Probabilistic top-k query processing in distributed sensor networks,” Proc. Int. Conf. on ICDE, pp.585–588, 2010.
[16] Scalable Network Technologoes: Creaters of QualNet Network Simulator Software, <URL: http://www.scalable-networks.com/>.