# Efficient Top-k Query Processing in Mobile Ad Hoc Networks

Takahiro Hara, Ryo Hagihara, Yuya Sasaki, Masako Shinohara, And Shojiro Nishio
Department of Multimedia Engineering, Osaka University

In mobile ad hoc networks (MANETs), to acquire only necessary data items, it is effective that each mobile node retrieves data items using a top-k query, in which data items are ordered by the score of a particular attribute, and the query issuing mobile node acquires data items with the $k$ highest scores. In this paper, we propose a top-k query processing method in MANETs for reducing traffic and keeping the accuracy of the query result. In this method, each mobile node estimates data items with the $k$ highest scores and sets a part of those scores to the standard scores. When a mobile node transmits query and reply messages, it reduces the number of candidates that are included in the top-k result by referring to the standard scores. We also present results of some simulation studies to evaluate the performance of our proposed method. The simulation results show that our method can reduce traffic for processing top-k queries as well as keeping high accuracy.

Keywords: Mobile ad hoc network, top-k query, mobile computing, radio link disconnection

## 1. INTRODUCTION

Due to recent advances in radio communication and computer technologies, there has been increasing interest in mobile ad hoc networks (MANETs), which are constructed only by mobile nodes (Baker, et al. [1982] and Broch, et al. [1998]). In MANETs, every mobile host plays the role of a router, and relays communication packets even if the source and the destination mobile nodes are not in the communication range of each other. Since no special infrastructure is required, in various fields such as rescue operations, military affairs, and sensing systems, many applications are expected to be developed in MANETs. A typical application in MANETs is information sharing among mobile users engaged in a collaborative work (e.g., rescue operations and military affairs). In such applications, mobile users generally share the information on the task allocation and progress of each member for improving efficiency of the collaborative work. Therefore, it often happens that mobile nodes issue a query to acquire data of their interests. Since the communication bandwidth and battery capacity of each node are limited in MANETs, it is important to acquire only necessary data items to reduce data traffic.

For this aim, a promising approach is that each mobile node retrieves data items using a top-k query, in which data items are ordered by the score of a particular attribute and the mobile node that retrieves data items (*the query issuing node*) acquires ones with the $k$ highest scores (*top-k result*). There are two naive manners to perform top-k query processing. The first manner is as follows. A query issuing node floods all mobile nodes in the entire network with a query message, each mobile node that receives the query transmits its own data items with the $k$ highest scores (*local top-k result*), and then, the query issuing node receives the data items (*query result*). In this manner, though all data items included in the top-k result are always in the query result (if network partitioning and packet losses do not occur), many data items that are not included in the top-k result are also transmitted to the query issuing node, which increases unnecessary traffic. Moreover, the increase of traffic often causes packet loses in a real situation, thus, the accuracy of the query result may decrease.

This research was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (18049050) and for Scientific Research (S)(21220002) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and by Mobile Wireless Foundation of Mobile Radio Center.
Authors' addresses: Takahiro Hara, Ryo Hagihara, Yuya Sasaki, Masako Shinohara, and Shojiro Nishio, Department of Multimedia Engineering, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan.

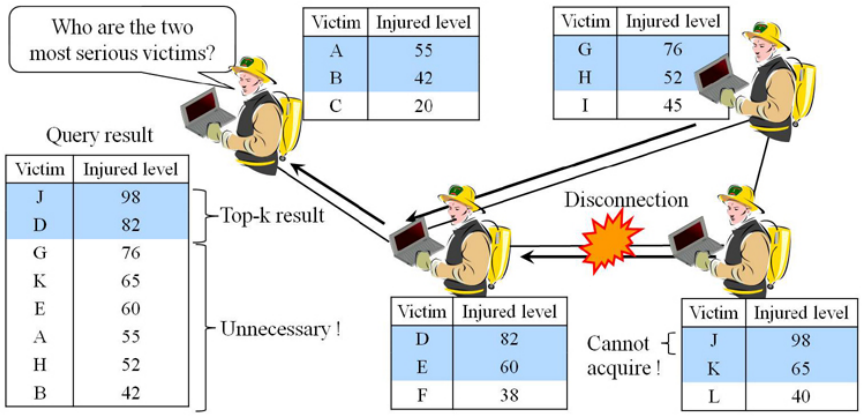



Figure. 1: Example of top-k query processing in MANETs

Figure 1 shows an application example of a rescue effort at a disaster site where the communication infrastructure, e.g. Internet, has broken down. Each rescuer manages information on victims and their injuries and searches for several victims with serious injuries to evacuate them with limited resources such as ambulances and medicines. When the rescuer at the top-left corner wants to find the two most seriously injured victims in the entire area, he performs a top-k query where $k$ is set to 2. Here, if each rescuer sends back the information on the two most injured victims in his responsible region, the query issuing rescuer acquires information on more victims than necessary.

The second nave manner is that each mobile node that receives the flooded query sends back a fixed number ($l$) of data items with the $l$ highest scores among its own data items. Note that the first manner is a special case ($l = k$) of the second manner. In the second manner, if $l$ is set as a small value (e.g., $l = 1$), the traffic can be reduced while the accuracy of the query result decreases (in particular, when $k$ is large). On the contrary, if $l$ is set as a large value ($l = k$), the same problem happens as the first manner.

As mentioned above, the two nave manners do not work well in MANETs. Thus, we need another approach to efficiently process top-k queries in MANETs. Moreover, since mobile nodes move freely in MANETs, connection and disconnections of radio links between mobile nodes frequently occur. If radio links between mobile nodes are disconnected during the execution of a top-k query, the query issuing node might not be able to acquire all data items that are included in the top-k result. This means that the accuracy of the query result decreases. In the example shown in Figure 1, if the radio link between the mobile nodes at the center and the bottom-right corner is disconnected, the query issuing rescuer cannot acquire information on victims managed by the mobile node at the bottom-right corner. Thus, we need some mechanism to cope with the network topology change during the execution of a top-k query.

In this paper, we propose a method for top-k query processing in MANETs for reducing traffic and also keeping high accuracy of the top-k result, i.e., guaranteeing to acquire all data items included in the top-k result when network partitioning and packet losses do not occur. In this method, each mobile node estimates data items with the $k$ highest scores and sets a part of those scores to the *standard scores*. When a mobile node transmits query and reply messages, it reduces the number of candidates of data items that are included in the top-k result by referring to the standard scores. Moreover, if a mobile node detects a disconnection of a radio link during the transmission of the reply message, it searches for an alternative path to transmit the reply message to the query issuing node. We also present simulation results to evaluate the performance of our proposed method.

The remainder of this paper is organized as follows: In section 2, we present some related works. In section 3, we propose a message processing method for top-k query. In section 4, we

show the results of the simulation experiments. Finally, in section 5, we summarize this paper. We note that some of the results of this paper have been reported in (Hagihara et al. [2009]).

## 2. RELATED WORKS

To the best of our knowledge, there have been no studies that address the issue of top-k query processing in MANETs before us (Hagihara et al. [2009]). In this section, we present some conventional studies on top-k query processing and data retrieval, which are related to our work. We only focus the three research fields: peer-to-peer (P2P) networks, wireless sensor networks (WSNs), and MANETs. While these three fields assume different types of networks, i.e., wired (P2P networks) or wireless (WSNs and MANETs) and fixed (P2P networks and WSNs) or mobile (MANETs), all of them have some similar characteristics such as multihop communication and limited knowledge obtained only from neighbor nodes.

### 2.1 Top-k Query Processing in P2P Networks

In the research field of unstructured P2P networks, various strategies for top-k query processing have been proposed. Kalnis et al. [2006] proposed a method in which a query issuing node floods all nodes with a query message that includes information on the number of requested data items ($k$). Then, each node that receives the query message orders its own data items by score, and transmits data items with $k$ highest scores. This method corresponds to the first naive manner described in section 1, thus, transmits data items that are not included in the top-k result, leaving the problem of the increase in unnecessary traffic unsolved. Balke et al. [2005] proposed a method in which a query issuing node acquires the data item with the highest score by transmitting a query message and continues this procedure $k$ times to acquire the top-k result. This method minimizes the number of transmitted data items but takes very long time to acquire the top-k result. Since the network topology dynamically changes in MANETs, it is not effective to apply this method. Akbarinia et al. [2006] proposed a method in which each node selectively transmits query and reply messages to its neighboring nodes in the P2P network in order to reduce the network traffic. This method is different from our proposed method because an environment where every message is transmitted by unicast and every two nodes can communicate with each other anytime is assumed. Matsunami et al. [2005] proposed a method in which each node determines the number of nodes to which the query message is forwarded in order to reduce reply messages. This method does not guarantee to acquire all data items included in the top-k result. On the contrary, our proposed method guarantees that the query issuing node can acquire all data items included in the top-k result if network partitioning and packet losses do not occur.

### 2.2 Top-k Query Processing in WSNs

In the research field of WSNs, some studies that aim at prolonging the network lifetime by reducing energy consumption for processing top-k queries. Yazti et al. [2007] proposed a method that predicts scores of data items included in the top-k result and reduces data items sent back to the query issuing node by using the predicted $k$-th highest score. Silberstein et al. [2006] proposed a method that utilizes the temporal correlation among sensor readings and avoid sensor readings from being sent to the query issuing node if those readings can be predicted from the past observations. These methods aim at reducing data items (sensor readings) sent back to the query issuing node by predicting the scores, which cannot guarantee that the query issuing node acquires all data items included in the top-k result.

### 2.3 Top-k Query Processing in WSNs

In the research field of MANETs, assuming that each mobile node retrieves data items by flooding query messages, many strategies for reducing traffic caused by flooding have been proposed. Mase et al. [2001] and Pagani et al. [1999] proposed methods in which each node creates a cluster with its neighboring mobile nodes, and only the cluster heads of clusters and mobile nodes that connect two clusters broadcast a query message in order to reduce traffic for data retrieval.

These methods are different from our proposed method because these methods aim at reducing traffic by reducing the number of broadcast messages, whereas our method reduces traffic by not transmitting data items that are not included in the top-k result.

Klemm et al. [2003] proposed a method that reduces traffic by reusing the past query results. In this method, each mobile node caches the results of queries issued by itself and other nodes, and answers queries (returns the cached data items) even if it is not the owner of the target data items when it receives queries to the data items. This method is different from our method because it assumes an environment where each mobile nodes can reuse the past query results. Shinohara et al. [2007] proposed a method that reduces traffic for answering queries by collecting multiple requests and constructing an efficient multicast tree for these requests. This method is different from our method because it assumes an environment where queries have some deadlines and can wait for a while (by the deadline) until receiving the query results (data items).

## 3. TOP-K QUERY PROCESSING METHOD

In this section, we first describe the design policy of our proposed method and our assumed environment. Then, we describe our proposed top-k query processing method.

### 3.1 Design Policy

In MANETs, mobile nodes connect to other mobile nodes by radio links, and the connection and disconnection of radio links between mobile nodes frequently occur due to the movement of mobile nodes. Thus, the query issuing node should acquire a query result in a short time to avoid disconnections of radio links during the query execution. For this aim, we design our method to process a query by just one round of message transmissions, i.e., flooding the network with a query message and collecting replies. Moreover, since mobile nodes consume the limited communication bandwidth for data transmission, packet loss and packet retransmission may occur when a network is congested, which result in some data items being not transmitted. Thus, the number of data items transmitted by each mobile node should be reduced as much as possible. Our proposed method aims at achieving this by attaching a small amount of extra information to the query message and reducing candidates of data items included in the query result.

In addition, the network topology may change during the query execution even if we adopt the above policy (one round of message transmissions). In this case, the mobile node that detects a disconnection should properly transmit its own reply message to the query issuing node along an alternative path to prevent the accuracy of the query result from decreasing.

### 3.2 System Model

The system environment is assumed to be a middle scale MANET in which a few dozens of mobile nodes retrieve data items held by themselves by using a top-k query. The detail of the system model is as follows:

- ✦ 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 $M = M_1, M_2, ..., M_m$, where $m$ is the total number of mobile nodes and $M_i (i = 1, ..., m)$ is a node identifier. These nodes move freely, thus, the network topology dynamically changes and sometimes network partitioning occurs.
- ✦ Data are handled as a collection of data items. We assign a unique *data identifier* to each data item located in the system. The set of all data items is denoted by $D = D_1, D_2, ..., D_n$, where $n$ is the total number of data items and $D_j (j = 1, ..., n)$ is a data identifier. Each data item consists of values of multiple attributes. For the purpose of simplicity, we assume that all data items are of the same size, $|D|$.
- ✦ Each data item is held by a particular mobile node (owner), and each mobile node can own multiple data items. For the purpose of simplicity, data items are not updated. It is also assumed that mobile nodes do not replicated data items held by other nodes.

✦ The query issuing node transmits a query message with the query condition and acquires data items with the $k$ highest scores among all data items held by mobile nodes in the entire network. A data-score is calculated from the query condition and the scoring function. Here, we do not place any restrictions on scoring functions since our proposed method is independent of them.

### 3.3 Proposed Method

Our proposed top-k query processing method aims at reducing traffic for query processing by attaching a small amount of extra information to the query message and reducing candidates of data items included in the query result. More specifically, each mobile node estimates data items with the $k$ highest scores and sets a part of those scores ($N$ scores) to the standard scores. When a mobile node transmits query and reply messages, it reduces the number of candidates that are included in the top-k result by referring to the standard scores. In the following, we describe the detail of our proposed method.

3.3.1 *Determining the Standard Scores.* As mentioned above, in our proposed method, each mobile node estimates data items with the $k$ highest scores and sets $N$ of them to its standard scores. Each standard score $B(i, j)$ $(1 \leqslant j \leqslant N)$, calculated by mobile node $M_i$, guarantees that there are at least $k_j/N$ $(1 \leqslant j \leqslant N)$ data items whose scores are equal to or larger than $B(i, j)$ on the path from the query issuing node to $M_i$. In other words, the query issuing node can acquire at least $k_j/N$ $(1 \leqslant j \leqslant N)$ data items by transmitting data items whose scores are equal to or larger than $B(i, j)$ from mobile nodes on the path from the query issuing node to $M_i$.

3.3.2 *Transmission of Query Messages.* To reduce the number of candidates of data items that are included in top-k result in our method, each mobile node attaches its own standard scores, to a query message to its neighboring nodes. Then, mobile nodes that receive the query message update the standard scores. The following is the list of elements composing a query message. Here, a mobile node that transmits and receives a query message is simply called a *query transmitter* and a *query receiver*, respectively.

✦ *Query issuing node ID*: The node identifier of the query issuing node.
✦ *Request number*: The number of data items, $k$, which the query issuing node requests.
✦ *Query condition*: The condition specified by the query issuing node.
✦ *Standard score list*: The list of standard scores determined by the query transmitter.
✦ *Number of hops*: The path length (hop count) from the query issuing node to the query transmitter.
✦ *Query transmitter ID*: The node identifier of the query transmitter. This is used to recognize the parent of the query receiver.
✦ *Parent ID*: The node identifier of the mobile node that transmits a query message to the query transmitter. This is used to recognize the child of the query receiver.

The procedures of the query issuing node, $M_p$, and mobile nodes that receive a query message are as follows:

(1) $M_p$ specifies the number of requested data items $k$ and the query condition. Then, $M_p$ calculates the scores of its own data items from the query condition using a scoring function and initializes its standard scores as follows:

$$B(p, j) = S(p, \frac{k}{N}j)(i \leqslant j \leqslant N) \quad (1)$$

Here, $S(i, h)$ denotes the $h$-th highest score among the scores calculated by $M_i$. Specifically, the standard scores of $M_p$ are the every $k/N$-th scores calculated by $M_p$.

(2) $M_p$ transmits a query message to its neighboring mobile nodes. In the query message, the query issuing node ID is set as $M_p$, the request number is set as $k$, the query condition is set as the specified condition, the standard score list is set as $B(p, j)$ $(1 \leqslant j \leqslant N)$, and the number of hops is set as 0. In addition, the query transmitter ID is set as $M_p$, and parent ID is set as empty.

(3) If a mobile node, $M_q$, receives the query message that already has received, the procedure goes to step 5. Otherwise, if $M_q$ firstly receives the query message, it sets the query transmitter to its parent and stores a pair of the number of hops and the query transmitter ID. $M_q$ calculates the standard scores for its own data items in the same way as $M_p$ did in step 1. Then, $M_q$ updates the standard scores as the $N$ highest scores among the standard scores in the received queries and those calculated for its own data items. Specifically, it sets its own standard scores as follows:

$$B(p, j) = Max(S(p, \frac{k}{N}a), L(b))(i \leqslant j \leqslant N) \tag{2}$$

Here, $L(l)$ $(1 \leqslant l \leqslant N)$ denotes the $l$-th highest score listed in the standard score list of the received query message. The initial values of $a$ and $b$ are equal to 1. $M_q$ calculates the next standard score corresponding to $j$ while incrementing $a$ by 1 if $B(q, j)$ is equal to $S(q, ka/N)$, i.e., $S(q, ka/N) \geqslant L(b)$, otherwise incrementing $b$ by 1. Note that $M_q$ guarantees that, for each j $(1 \leqslant j \leqslant N)$, there are more than data items with scores equal to or larger than $M_q$'s standard score $B(q, j)$ $(1 \leqslant j \leqslant N)$ among data items held by itself and mobile nodes along the path where the query message was transferred. Then, the procedure goes to step 4.

(4) $M_q$ sets the query transmitter and parent IDs as $M_q$ and $M_q$'s parent, respectively. In addition, $M_q$ updates the standard score list to $B(q, j)$ $(1 \leqslant j \leqslant N)$ and increments the number of hops by 1. $M_q$ transmits the query message to its neighboring mobile nodes. Goes back to step 3.

(5) $M_q$ sets the query transmitter to its child if the parent ID in the query message is $M_q$. Otherwise, $M_q$ stores a pair of the number of hops and the query transmitter ID and updates its own standard scores $B(q, j)$ $(1 \leqslant j \leqslant N)$ in the same way as step 3.

In the transmission of the query message, each mobile node reduces the number of candidates of data items that are included in the top-k result by using the standard scores. Moreover, by using the query transmitter and parent IDs in the query message, each mobile node can recognize its parent and children in the query propagation tree whose root is the query issuing node. Each mobile node also recognizes the relationship between itself and the neighboring nodes who are neither its parent nor children by using the pair of the number of hops and the query transmitter ID.

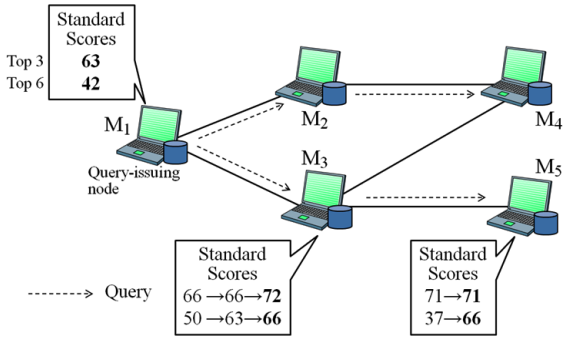



Figure. 2: Example of top-k query processing in MANETs)

Table I: Scores of data items

| **Data** | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ |
|---|---|---|---|---|---|
| 1 | 88 | 90 | 80 | 74 | 86 |
| 2 | 70 | 83 | 77 | 69 | 81 |
| 3 | 63 | 72 | 66 | 59 | 71 |
| 4 | 55 | 61 | 60 | 53 | 68 |
| 5 | 49 | 51 | 58 | 44 | 52 |
| 6 | 42 | 40 | 50 | 34 | 37 |
| 7 | 30 | 31 | 41 | 28 | 22 |
| ... | ... | ... | ... | ... | ... |

Figure 2 shows an example when $M_1$ retrieves data items with the six highest scores using two standard scores ($k = 6, N = 2$). In this figure, the lists in the balloons denote two standard scores, which each mobile node sets. The rightward side of the arrows in the balloons denotes the updated standard scores. Table 1 shows the scores of data items held by each mobile node, which are sorted by score (descending order). For example, when $M_2$ receives a query message from $M_1$, it compares standard scores for its own data items with the standard scores in the query message, updates its own standard scores as 72 and 63, and continues to transmit the query message. On the other hand, $M_3$ first set its own standard scores as 66 and 63 when it receives the query message from $M_1$, and then updates them as 72 and 66 when it receives the query message from $M_4$, which is neither its parent nor child.

3.3.3 *Transmission of Reply Messages.* Since each mobile node can reduce the number of candidates of data items included in the top-k result as described in section 3.3.2, it transmits a reply message attached with only these candidates. In addition, when a mobile node that receives a reply message finds certain data items are surely not included in the query result from the received information, it does not transmit those data items.

The following is the list of elements composing a reply message.

- *Query issuing node ID*: The node identifier of the query issuing node.
- *Threshold*: The $k$-th highest score estimated by the mobile node that transmits the reply message.
- *Reply list*: The list of the pair of the scores that are equal to or larger than the threshold and the data items with the corresponding scores.

The procedure of each mobile node that has finished the transmission of query messages is as follows:

(1) A mobile node, $M_s$, which has no children, transmits a reply message to its parent. In the reply message, the query issuing node ID is set as $M_p$. The threshold is set as $B(s, N)$ because as described in section 3.3.2, it is guaranteed that at least $k(= kN/N)$ data items with scores equal to or larger than the standard score $B(s, N)$ exist on the path from the query issuing node to $M_s$. The reply list contains a list of pairs of the scores that are equal to or larger than the threshold $B(s, N)$ and the data items with the corresponding scores.

(2) When a mobile node, $M_t$, receives a reply message, $M_t$ updates the threshold with the higher score between the threshold in the received reply message and $B(t, N)$. $M_t$ deletes scores, which are smaller than the threshold, and data items with the corresponding scores from the reply list in the received reply message. Instead, $M_t$ inserts scores of its own data items, which are equal to or larger than the threshold, and data items with the corresponding scores to the reply list. Then, if the number of data items contained in the reply list becomes larger than $k$, $M_t$ updates the threshold with the $k$-th score in the reply list and deletes scores, which are smaller than the new threshold, and data items with the corresponding scores in the reply list. When $M_t$ receives reply messages from all its children, it transmits the updated

reply message to its parent. This procedure repeats until the query issuing node receives the reply message from all its children.

As mentioned above, along the path from each leaf node (node having no children) in the query propagation tree to the query issuing node, at most $k$ data items which have possibility to be included in the query result are sent back to the query issuing node. In other words, each mobile node does not transmit data items that are surely not included in the top-k result using the threshold. Therefore, our proposed method can reduce traffic for processing a top-k query while keeping high accuracy of the query result. In particular, if network partitioning and packet loses do not occur, our method guarantees that the query issuing node can acquire all $k$ data items included in the query result. Figure 3 shows an example of the procedure in which each mobile

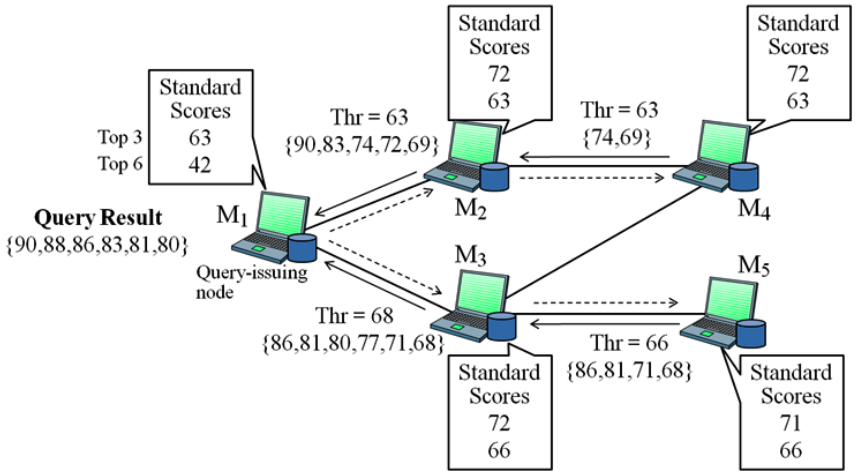



Figure. 3: Reply message transmission

node transmits a reply message. In this figure, the list in each balloon denotes the standard score list of the corresponding mobile node after the query message transmission process in Figure 2. The value denoted by "Thr=" attached to each radio link shows the threshold in the reply message. The list of scores which is also attached to each radio link denotes the scores contained in the reply list in the reply message. In this figure, $M_4$, which has no children, sets the threshold as the second standard score (63) and the reply list in the reply message to contain the scores that are equal to or larger than 63 and data items with the corresponding scores. Then, $M_4$ transmits the reply message to $M_2$. Because the second standard score of $M_2$ (63) is same as the threshold in the received reply message from $M_4$, $M_3$ updates the reply list in the reply message by inserting its own scores that are equal to or larger than the threshold (63) and data items with the corresponding scores. On the other hand, $M_3$ first sets the threshold as 66, but the number of data items with scores equal to or larger than the threshold becomes 7, which is larger than $k$ (=6). Therefore, $M_3$ updates the threshold with the 6-th score (68) and deletes the 7-th score (66) and the corresponding data item from the reply list. It then transmits the reply message to $M_1$.

3.3.4 *Detection of Radio Link Disconnection.* In MANETs, the network topology dynamically changes due to the movement of mobile nodes. If a mobile node cannot transmit a reply message to its parent because of a disconnection of the radio link between these nodes, the accuracy of the query result decreases. Therefore, when a mobile node detects a disconnection of the radio link to its parent, it searches for an alternative path through other nodes to transmit the reply message along the discovered path.

The following is the procedure when a mobile node, $M_u$, detects a radio link disconnection.

(1) When $M_u$ detects a disconnection to its parent, it transmits a reply message to a neighboring mobile node except for $M_u$'s children. If there are multiple candidates, $M_u$ selects the mobile node with the smallest number of hops, which can be known from the procedure described in section 3.3.2, to reduce traffic and delay as much as possible. On the other hand, if $M_u$ detects a disconnection to its child, it deletes the information on its child and transmits a reply message to its parent when it receives reply messages from all its children except for the disconnected one.

(2) When a mobile node, $M_v$, receives a reply message from a mobile node which is not $M_v$'s child, it transmits a reply message to its parent as follows. If $M_v$ is waiting for reply messages from its children, it treats the received reply message from the (non-child) node in the same way as $M_v$'s children. Otherwise, if $M_v$ has already transmitted the reply message, it updates the threshold in the same way as the procedure described in section 3.3.3. Then, $M_v$ deletes the scores that are smaller than the updated threshold and data items with the corresponding scores in the newly received reply message and transmits the remaining scores and data items as the new reply message to its parent.

Here, if all $M_u$'s neighboring nodes are its children, the above procedure does not work. This is because a reply message transmitted to its child will be immediately sent back to $M_u$, i.e., a message loop would occur. To avoid this problem, $M_v$ constructs an alternative path using a path-search message. The following is a list of elements composing a path-search message.

- ✦ *Transmitter node ID*: The node identifier of the mobile node that transmits a path-search message.
- ✦ *Path list*: The list of node identifiers of mobile nodes along the path where the path-search message is transmitted.

The procedure of $M_u$ searching for an alternative path is as follows:

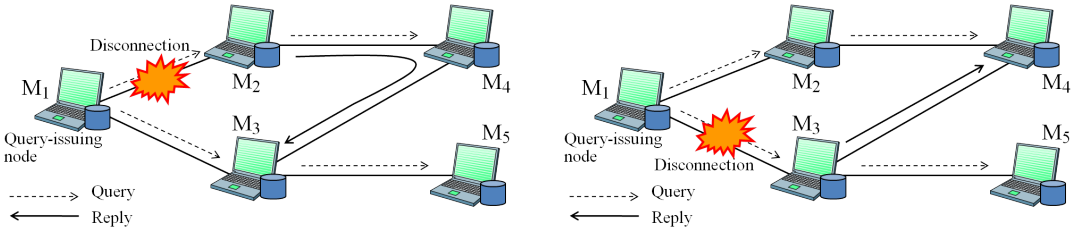
(1) $M_u$ transmits a path-search message to all its children. In this message, the transmitter node ID is set as $M_u$ and the path list is set as only $M_u$.

(2) When a mobile node, $M_v$, receives a path-search message, it checks whether a neighboring mobile node, whose parent is not the transmitter node in the received message, exists or not. If such a mobile node exists, it is obvious that the path from the node to the query issuing node does not include the disconnected radio link. Thus, $M_v$ adds the path information from $M_v$ to the query issuing node via the found node to the path list in the received message from $M_u$ and sends it back to $M_u$.

(3) When $M_u$ receives the path list from $M_v$, it transmits the reply message along the newly discovered path, and $M_v$ processes the reply message in the same way as described in section 3.3.3. Otherwise, if such a mobile node does not exist, $M_v$ inserts itself into the path list in the received path-search message and transmits the message to its children.

In the procedure of detecting a radio link disconnection, each mobile node transmits a reply message through the discovered path even if the radio link to its parent is disconnected. Therefore, the query issuing node can receive the reply message from all mobile nodes in the entire network (if network partitioning and packet loses do not occur).

Figure 4 shows an example of the procedure in which a mobile node detects a radio link disconnection and finds an alternative path. If the radio link between $M_1$ and $M_3$ is disconnected (Figure 4(a)), $M_3$ transmits the reply message to $M_4$, which is not a child of $M_3$. Meanwhile, if the radio link between $M_1$ and $M_2$ is disconnected (Figure 4(b)), $M_2$ transmits a path-search message to $M_4$, which is a child of $M_2$. Since $M_4$ finds the neighboring node $M_3$, whose parent is not $M_2$, it transmits the discovered path ($M_2 \rightarrow M_4 \rightarrow M_3$) to $M_3$. Afterward, $M_3$ transmits the reply message along the discovered path.




(a) Radio link disconnection between $M_1$ and $M_2$ (b) Radio link disconnection between $M_1$ and $M_3$

Figure. 4: Procedure of detecting radio link disconnection

## 4. 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 (http://www.scalable-networks.com).

### 4.1 Simulation Model

In our simulations, we assume a medium scale MANET in which mobile users engaged in a collaborative work such as rescue operations share information. Specifically, 50 mobile nodes, $M_1$,..., $M_{50}$, exist in a two-dimensional field of 600 [m] × 600 [m]. The initial position of each mobile node is randomly determined. Each mobile node moves according to the random waypoint model (Camp et al. [2002]) where the movement speed is randomly determined from 1 to 5 [m/sec] and the pause time is 60 [sec]. We assume two different cases regarding data communication. In the first case, each mobile node transmits messages and data items using an IEEE 802.11b device whose data transmission rate is 11 [Mbps]. We assume that packet losses occur due to radio interference. We denote this case as "real case." In the second case, it is assumed that the data transmission rate is sufficiently high and no packet losses occur. We denote this case as "ideal case." In the both cases, the transmission power of each mobile node is determined so that the radio communication range becomes about 100 [m].

Each mobile node holds 100 data items. The size of all data items is 1 [KB]. The scores of data items held by mobile nodes are skewed. Specifically, we divide mobile nodes into four groups $G_0$, $G_1$, $G_2$, and $G_3$, where node $M_i$ ($i$=1,...,50) is classified into $G_{(i \bmod 43)}$. In $G_x$ ($x$=0,1,2,3), scores of data items held by mobile nodes are randomly determined within the range from $25x$ to $25(x+1)$. We assume that data updates periodically occur at interval 2,000 [sec]. The time when each data item is updated at the first time is randomly determined within the range from 0 to 2,000 [sec]. Each query issuing node waits for 30 [sec] to collect replies, i.e., the query deadline is 30 [sec]. When each mobile node on the query propagation path cannot received an ACK from its parent for 1 [sec] after sending the reply, it retransmits the reply and repeats this at most five times. If all the retransmissions result in failure, the node tries to find an alternative path by the procedure described in section 3.3.4.

The number of requested data items, $k$, is basically 100 and is varied from 1 to 100 in some simulation experiments. The number of standard scores, $N$, is varied from 1 to $k$ in the first simulation experiments in order to examine its impact on the performance. In other experiments, the optimal values of $N$ which are obtained through the first experiments are used.

We compare the performance of our proposed method with that of a nave method where each mobile node that receives a query message sends back a fixed number, $R$, of its holding data items with the $R$ highest scores. In the nave method, if each mobile node finds that the total number of data items received from all its children and its own data items with the $R$ highest scores becomes larger than $k$, it only sends $k$ data items with the highest scores among those data items to its parent. We examine three different cases for $R$; $R$= $\lceil k/50 \rceil$, $\lceil k/2 \rceil$ , and $k$ (simply

denoted by $R = k/50, k/2$, and $k$).

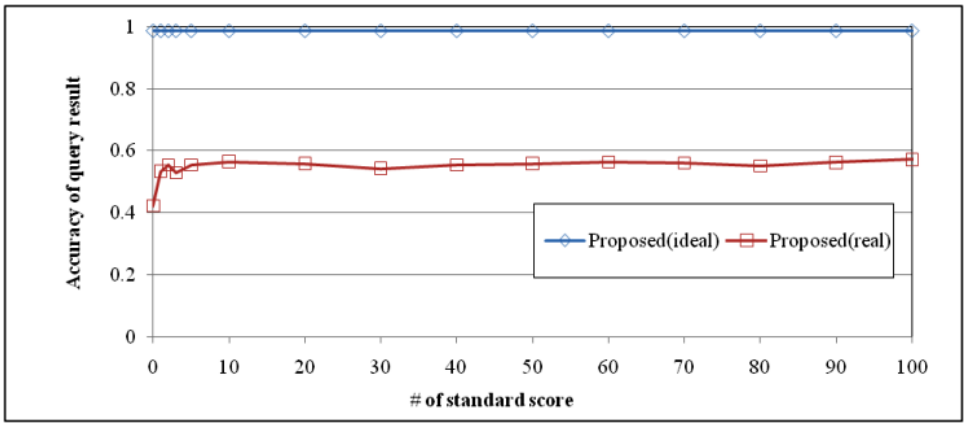
In the above simulation model, we evaluate the following criteria. These criteria are defined as the average for all 50 queries issued during the simulation time.

- *Accuracy of query result*: The ratio of the number of acquired data items that are included both in the query result and the top-k result to the number of requested data items performed during the simulation.
- *Traffic*: The summation of the message size of queries, replies, and path-searches transmitted during the simulation.

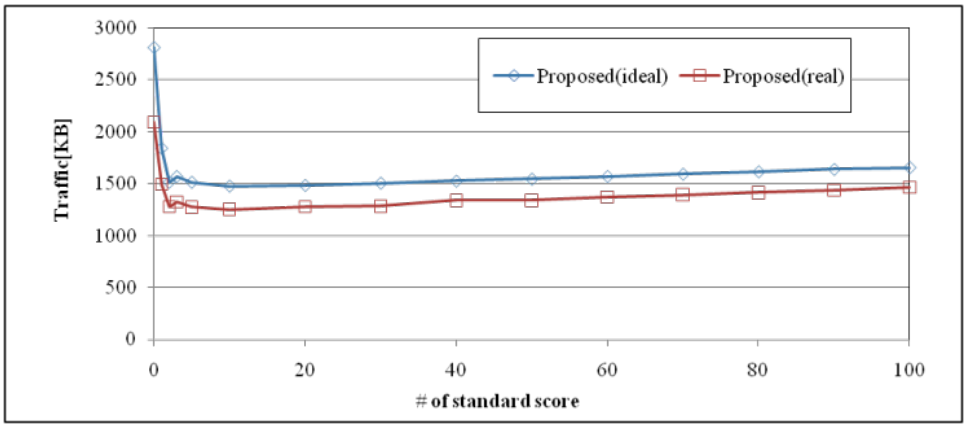
## 4.2 Simulation Results

In this subsection, we first examine the impact of the number of standard scores, $N$, on the performance of our proposed method. Then, we compare the performance of our method with the naive method.

4.2.1 *Impact of the Number of Standard Scores.* We examine the impact of the number of standard scores, $N$, on the performance metrics in our proposed method. We first measure the accuracy of query result and traffic of our method when $k$ is fixed as 100 and $N$ is varied from 1 to 100.




(a) Accuracy of query result




(b) Traffic

Figure. 5: Effect of the number of standard scores, $N$

Figure 5 shows the simulation result for the two cases, ideal (without packet loses) and real (with packet loses), regarding data communication. In both graphs, the horizontal axis indicates the number of standard scores, $N$. The vertical axis indicates the accuracy of query result in the case of (a) and the traffic in the case of (b).

From Figure 5(a), it is shown that in the ideal case without packet loses, our proposed method achieves almost 100% of accuracy independent of the number of standard scores. This is because our proposed method is designed to guarantee that all data items included in the top-k result are acquired by the query issuing node if network partitioning and packet loses do not occur. The accuracy of query result does not reach to 1 due to the existence of network partitioning (but it rarely happens in this simulation setting). It is also shown that in the real situation with packet loses, the accuracy of query result is much lower than that in the ideal situation. Moreover, as $N$ increases, the accuracy of query result first increases and then become almost constant. This is due to the impact of packet loses, i.e., the query issuing nodes cannot acquire some data items included in the top-k results due to packet loses. In particular, when $N$ is very small, our proposed method cannot sufficiently reduce the candidate of data items included in the top-k result, which causes the increase of traffic and packet loses (we can confirm this fact from Figure 5(b)).

From Figure 5(b), as $N$ increases, the traffic drastically decreases at first, and then, it increases from a certain point. As $N$ increases, the traffic produced by query messages increases because the size of the query message is proportional to $N$. Meanwhile, the traffic produced by reply messages decrease because each mobile node effectively reduces the candidates of data items included in the top-k result. As a result, when $N$ is very small, our proposed method cannot sufficiently reduce the candidates of data items included in the top-k result, thus, the total traffic becomes very high. On the other hand, even if $N$ gets much larger, the impact of traffic reduction using standard scores becomes almost constant. Therefore, due to the impact of the increase of traffic produced by query messages, the total traffic slightly increases. Compared between the two cases, the real case with packet loses produces smaller traffic than the ideal case because some data items cannot be transmitted due to network congestions.

Next, we examine the optimal values of N for various cases of $k$ in the same way as in the case of $k$=100. Specifically, we give higher priority to the traffic reduction when determining the optimal value of $N$. If there are multiple values of $N$ that achieve the lowest traffic, we choose the one that achieves the highest accuracy of query result as the optimal value. Figure 6 shows

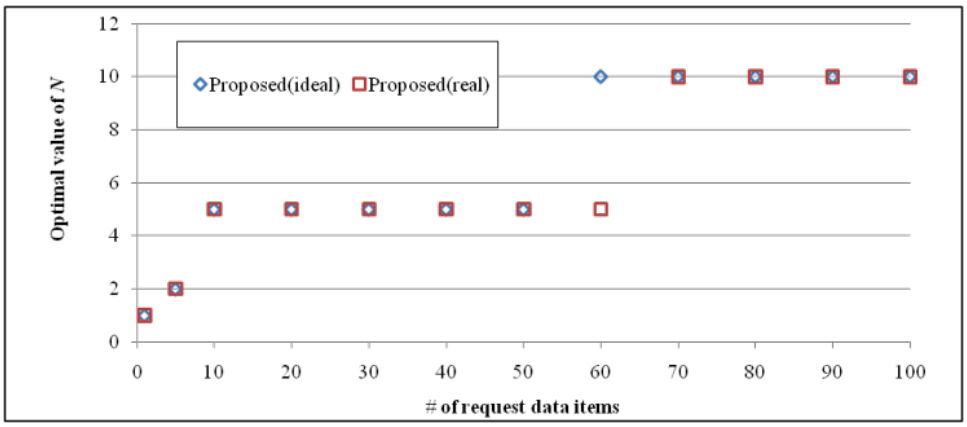
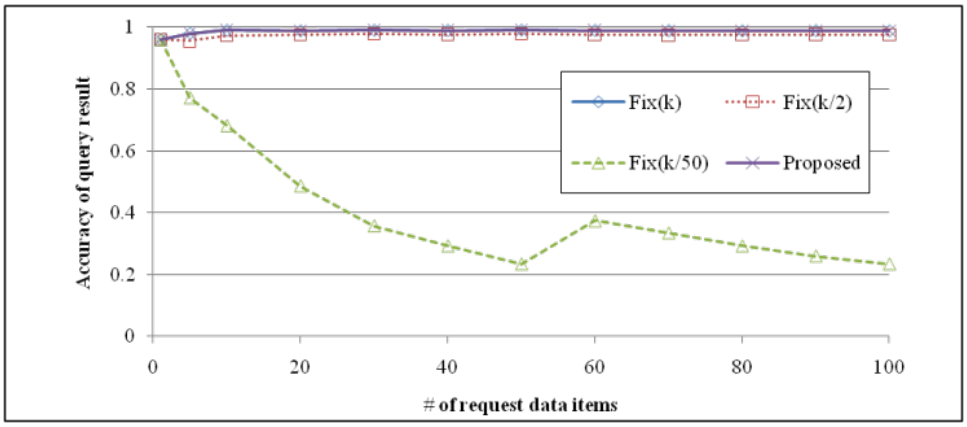


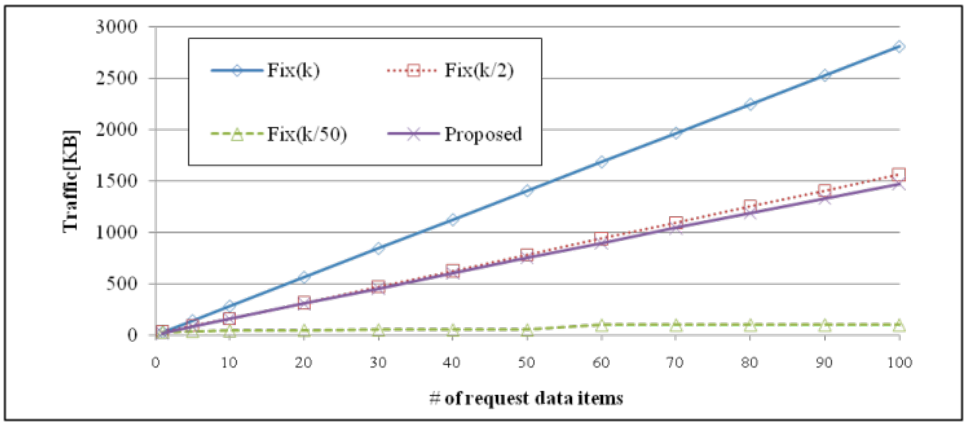

Figure. 6: Optimal values of $N$)

the examination result. In this graph, the horizontal axis indicates the number of requested data items, $k$, and the vertical axis indicates the optimal value of $N$ for the corresponding $k$.

From this result, in both ideal and real cases, as $k$ increases, the optimal value of $N$ increases stepwise. As $k$ increases, more standard scores are needed to reduce the candidates of data items included in the top-k result, thus, the optimal value slightly increases. Compared with the two cases, only when $k$=60, the optimal value of $N$ in the real case is lower than that in the ideal case. The traffic for reply messages in the real case is lower than that in the ideal case due to packet loses, and thus, the impact of the traffic for query messages (i.e., the impact of $N$ on the traffic) becomes relatively higher. Therefore, the increase of the optimal value of $N$ in the real case is a little bit slower than that in the ideal case.




(a) Accuracy of query result




(b) Traffic

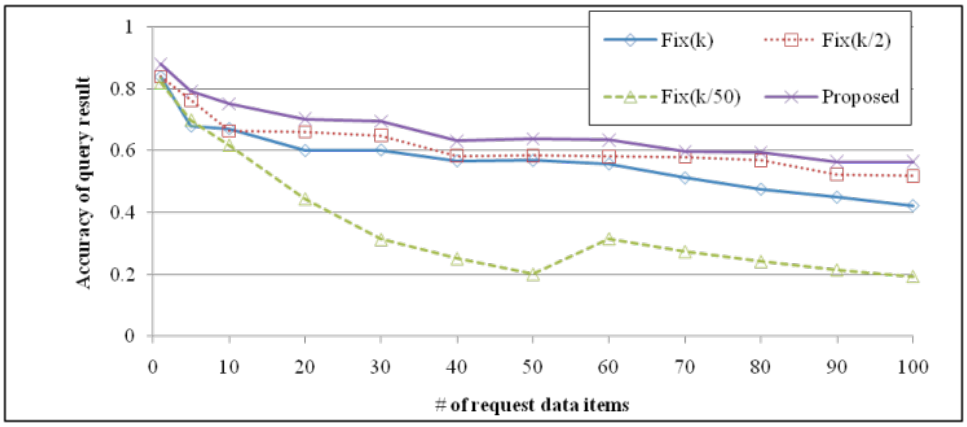
Figure. 7: Performance comparison (real case)

4.2.2 *Comparison between Our Method and the Naive Method.* Next, we evaluate the performance of our proposed method in comparison with the three cases of the native method, $R = k/50$, $k/2$, and $k$. Figures 7 and 8 respectively show the simulation results for the two cases, ideal (without packet loses) and real (with packet loses), regarding data communication. In all graphs, the horizontal axis indicates the number of requested data items, $k$. The vertical axis indicates the accuracy of query result in the case of (a) and the traffic in the case of (b). "Fix($k$)" denotes the nave method where $R = k$, and so on. From Figure 7(a), in the ideal case without packet loses, both of our method and the naive method except for the case of Fix($k/50$) achieves nearly 100% of accuracy of query result. As $k$ increases, the accuracy of query result in Fix($k/50$)

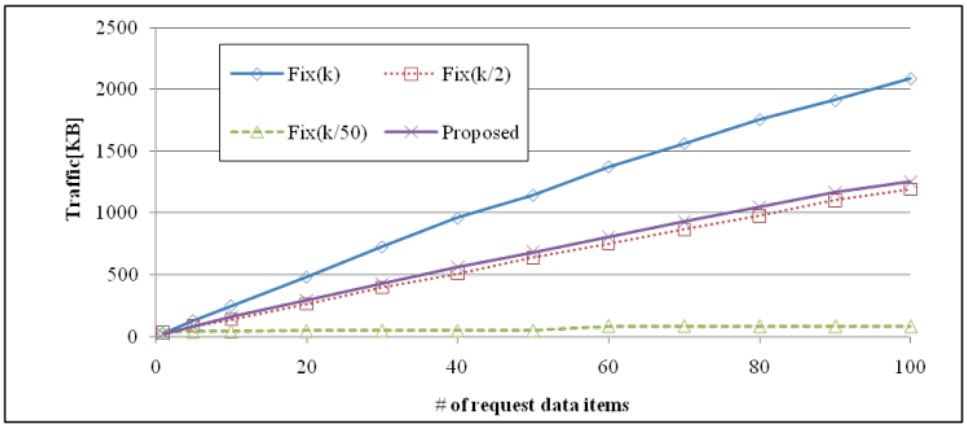
decreases. This is because in Fix($k$/50), each mobile node sends back only one (where $k \leqslant 50$ ) or two (where $k > 50$) data items among its own data items, thus, more data items that should be included in the top-k result cannot be transmitted to the query issuing node as $k$ increases. The accuracy of query result increases when $k$=60 because the number of data items sent back to the query issuing node increments where $k > 50$ (from one to two). Of the other three cases, our proposed method and Fix($k$) achieve slightly higher accuracy than Fix($k$/2).

From Figure 7(b), as expected, Fix($k$) produces very high traffic while Fix($k$/50) produces very low. Fix($k$/2) and our proposed method produce almost the same traffic. Compared between our method and Fix($k$), our method reduces about 47% of traffic.

To summarize the result shown in Figure 7, in the ideal case without packet loses, our proposed method achieves almost the same performance as Fix($k$/2). This fact shows that in the ideal case, the naive method in which each node sends back a fixed number ($R$) of its own data items can achieve almost the same performance by appropriately choosing the value of $R$. However, it is not easy for systems or system administrators to choose an appropriate value of $R$ according to the system situation and query characteristics. On the other hand, as mentioned above, the performance of our method is not very sensitive to the number of standard scores, which is only a parameter that the system or system administrators have to set. Thus, it is expected that our method can constantly achieve good performance independent of the system situation and query characteristics.




(a) Accuracy of query result




(b) Traffic

Figure. 8: Performance comparison (real case)

Figure 8(a) shows that in the real case with packet loses, the accuracy of query result in all methods decreases compared with the ideal case due to the impact of packet loses. Moreover, as $k$ increases, the accuracy in all methods decreases because the increase of traffic causes packet loses more often. Of the four methods, our proposed method always achieves the highest accuracy. Fix($k$) achieves the second lowest accuracy because this method produces very high traffic (as shown in Figure 8(b)) and packet loses occur more often.

Figure 8(b) shows almost the same result as that in Figure 7(b). Compared between Fix($k/2$) and our proposed method, our method produces slightly higher traffic than Fix($k/2$). However, as shown in Figure 8(a), our method achieves higher accuracy of query result. In Fix($k/2$), since each node sends back its own data items with the highest $k/2$ scores (though intermediate nodes can filter out some unnecessary data items), network congestions and packet loses are likely to occur at the early stage of transmissions of reply messages. More specifically, even leaf nodes in the query propagation tree transmit many (i.e., $k/2$) data items, packet loses start to occur from the leaf nodes. On the other hand, since our method can reduce the candidates of data items included in the top-k result by using standard scores, leaf nodes tend to send a small number of its own data items. This means that our method utilizes the network bandwidth more efficiently than Fix($k/2$). Therefore, while our method produces slightly higher traffic than Fix($k/2$), it achieve higher accuracy.

## 5. CONCLUSIONS

### 5.1 Summary

In this paper, we proposed a method for top-k query processing in MANETs which not only reduces traffic but also keeps high accuracy of the query result. In our proposed method, each mobile node estimates data items with the $k$ highest scores and sets a part of the scores as the standard scores. When a mobile node transmits query and reply messages, it reduces the candidates of data items included in the top-k result by referring to the standard scores attached to those messages. Moreover, if a mobile node detects a disconnection of a radio link, it searches for an alternative path to transmit the reply message to the query issuing node.

The simulation results showed that our proposed method reduces the traffic for query processing by reducing the number of candidates of data items using the standard scores. In particular, in a realistic environment where packet loses occur, our proposed method outperforms a native method in which each mobile node sends back a fixed number of its own data items with high scores.

### 5.2 Future Work

We think the top-k query processing method proposed in this paper is the first step to address the issue of efficient top-k processing in MANETs. Finally in this subsection, we present our on-going projects on this issue and discuss the future directions.

5.2.1 *Further Traffic Reduction based on the Score Distribution Estimation.* The method proposed in this paper attaches a small amount of information to query and reply messages and reduces the candidates of data items included in the top-k result. This method can achieve both high accuracy of the query result (100% if network partitioning and packet loses do not occur) and traffic reduction. However, we expect that we can further reduce traffic for query processing by adopting some techniques to estimate the score distribution in the entire MANET. We are now addressing this issue in our on-going project and the preliminary result has been reported in (Sasaki et al. [2010]).

5.2.2 *Data Replication to Cope with Network Partitioning.* As mentioned, our proposed method achieves high accuracy of the top-k query result if network partitioning does not occur. However, in a real situation, network partitioning does occur and our proposed method cannot do anything when the network is partitioned, i.e., the accuracy of the query result drastically decreases. To

cope with network partitioning and keep high accuracy of top-k query result, data replication is a promising solution. While there have been a large number of studies that address data replication in MANETs (Hara et al. [2006], Padmanabhan et al. [2008]), we have to design different replication techniques that are suitable for top-k query processing. We are now addressing this issue in our on-going project and the preliminary result has been reported in (Hara et al. [2010]).

5.2.3 *Other Types of Queries in MANETs.* We also plan to address other types of queries in MANETs such as $k$-nearest neighbor search, skyline query, and keyword search. Because MANET has some characteristics that are much different from other conventional networks such as the Internet, we should design query processing techniques suitable for MANET for each of the different types of queries.

## 6. ACKNOWLEDGMENT

This research was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (18049050) and for Scientific Research (S)(21220002) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and by Mobile Wireless Foundation of Mobile Radio Center.

## REFERENCES

Akbarinia, R., Pacitti, E., AND Valduriez, P. 2006. Reducing network traffic in unstructured P2P systems using top-k queries. In *Distributed and Parallel Databases 19(2-3).*,67-86.

Baker, D.J., Wieselthier,J., AND Ephremides, A. 1982. A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network. In *Proceedings of ICC.*, 1982, 2F6.1-2F6.5.

Balke, W.-T., Nejdl, W., Siberski, W., AND Thaden, U. 2005. Progressive distributed top-k retrieval in peer-to-peer networks. In *Proceedings of ICDE.*, 2005. 174-185.

Broch, J., Maltz, D.A., Johnson, D.B.,Hu, Y.C. AND Jetcheva, J. 1998. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of MobiCom.*, 1998, 159-164.

Camp, T., Boleng, J., AND Davies, V. 2002. A survey of mobility models for ad hoc network research In *Wireless Communications and Mobile Computing 2(5)*, 483-502.

Hagihara, R., Shinohara, M.,Hara, T. AND Nishio, S. 2009. A message processing method for top-k query for traffic reduction in ad hoc networks, In *Proceedings of International Conference on Mobile Data Management*, 2009, 11-20.

Hara, T. AND Madria, S.K. 2006. Data replication for improving data accessibility in ad hoc networks. In *IEEE Transactions on Mobile Computing 5(11)*, 1515-1532.

Hara, T., Hagihara, R., AND Nishio, S. 2010. Data replication for top-k query processing in mobile wireless sensor networks. In *Proceedings of International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, 2010, 115-122.

Kalnis, P., Ng, W.S.,Ooi, B.C. AND Tan, K.-L. 2006. Answering similarity queries in peer-to-peer networks. In *Information Systems 31(1)*, 57-72.

Klamm, A., Lindemann, C., AND Waldhorst, O.P. 2003. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks, In *Proceedings of Vehicular Technology Conference*, 2003, 2758-2763.

Mase, K., Sato, T., Nakano, K., Sengoku, M., AND Shinoda, S. 2001. Efficient flooding schemes in mobile ad hoc networks, In *Proceedings of International Conference on Multi-Dimensional Mobile Communications*, 2001, 152-157.

Matsunami, H., Terada, T., AND Nishio, S. 2005. A query processing mechanism for top-k query in P2P networks. In *Proceedings of International Special Workshop on Databases for Next Generation Researchers*, 2005, 84-87.

Padmanabhan, P., Gruenwald, L., Vallur, A., AND Atiquzzaman, M. 2008. A survey of data replication techniques for mobile ad hoc network databases. *VLDB Journal 17(5)*, 1143-1164.

Pagani, E., AND Rossi, G.P. 1999. Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks, *Mobile Networks and Applications 4(3)*, 175-192.

Sasaki, Y., Hagihara, R., Hara, T., Shinohara, M., AND Nishio, S. 2010. A top-k query method by estimating score distribution in mobile ad hoc networks. In *Proceedings of International Workshop on Data Management for Wireless and Pervasive Communications*, 2010, 944-949.

Silberstein, A., Munagala, K., AND Yang, J. 2006. Energy-efficient monitoring of extreme values in sensor networks, In *Proceeding of ACM SIGMOD Conference*, 2006, 169-180.

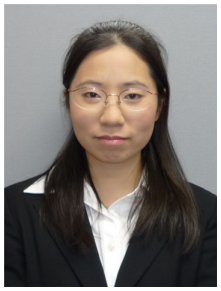
Shinohara, M., Hara, T., AND Nishio, S. 2009. A data transmission method using multicast in mobile ad hoc networks, In *Proceedings of International Conference on Mobile Data Management*, 2009, 232-237.

Yazti, D.Z., Andreou, P., Chrysanthis, P.K., AND Samaras, G. 2007. MINT views: Materialized in-network top-k views in sensor networks, In *Proceeding of International Conference on Mobile Data Management*, 2007, 182-189.

**Takahiro Hara** received the B.E, M.E, and Dr.E. degrees from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is an Associate Professor of the Department of Multimedia Engineering, Osaka University. He has published more than 100 international Journal and conference papers in the areas of databases, mobile computing, peer-to-peer systems, WWW, and wireless networking. He served and is serving as a Program Chair of IEEE International Conference on Mobile Data Management (MDM'06 and 10) and IEEE International Conference on Advanced Information Networking and Applications (AINA'09). He guest edited IEEE Journal on Selected Areas in Communications, Sp. Issues on Peer-to-Peer Communications and Applications. He served and is serving as PC member of more than 120 international conferences such as IEEE ICNP, WWW, DASFAA, ACM MobiHoc, and ACM SAC. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is an IEEE Senior member and a member of four other learned societies including ACM.

**Ryo Hagihara** received the Bachelor of Engineering degree and Master of Information Science and Technology degree from Osaka University, Osaka, Japan, in 2008, and 2010, respectively. His research interests include data search mechanisms in mobile computing environments.

**Yuya Sasaki** received the Bachelor of Engineering degree in 2009. Currently, He is a Master course student of the Department of Multimedia Engineering, Osaka University. His research interests include data search and replication mechanisms in mobile computing environments.



**Masako Shinohara** received the Bachelor of Engineering degree, and Master and Doctor of Information Science and Technology degrees from Osaka University, Osaka, Japan, in 2004, 2006, and 2009, respectively. Her research interests include energy-efficient data access and replication mechanisms in mobile ad hoc networks.

**Shojiro Nishio** received the B.E., M.E., and Ph.D. degrees from Kyoto University, Kyoto, Japan. He is currently a full professor in the Department of Multimedia Engineering, Osaka University, Japan. He has been serving as a Trustee and Vice President of Osaka University since August 2007. His areas of expertise in database systems include concurrency control, knowledge discovery, multimedia systems, and database system architectures for advanced networks such as broadband networks and mobile computing environment.
Dr. Nishio has co-authored or co-edited more than 25 books, and authored or co-authored more than 200 refereed journal or conference papers.
Dr. Nishio has served as a member of Program or Organizing Committees for more than 85 international conferences including VLDB, ACM SIGMOD, and IEEE ICDE. He served as the Program Committee Co-Chairs for several international conferences including DOOD 1989, VLDB 1995, and IEEE ICDE 2005.
He has served and is currently serving as an editor of several international journals including "IEEE Trans. on Knowledge and Data Engineering", "VLDB Journal", "ACM Trans. on Internet Technology", and "Data and Knowledge Engineering". He is a member of eight learned societies, including Senior members of ACM and IEEE.