Geoinformatica (2019) 23:591–620
https://doi.org/10.1007/s10707-019-00363-0

Efficient framework for processing top-k queries with replication in mobile ad hoc networks

Yuya Sasaki[1] · **Takahiro Hara**[1] · **Yoshiharu Ishikawa**[2]

Received: 13 October 2018 / Revised: 3 April 2019 / Accepted: 1 May 2019 /
Published online: 14 May 2019

Abstract
This article addresses the top-k query processing problem on mobile ad hoc networks (MANETs). Top-k query processing is common to retrieve only highly important data items. However, methods for top-k query processing are not enough efficient and accurate in MANET environments. For improving the efficiency and accuracy, replication is a promising technique that each node in MANETs replicates data items retained by other nodes into its storage. Therefore, we fully combine the top-k query processing with data replication. We propose a framework that efficiently processes top-k queries based on a new replication strategy. We develop new replication strategy *FReT* (topology-Free Replication for Top-k query). FReT determines near-optimal allocations of replicas. It advantages efficient top-k query processing from limited search area without maintenance costs even if mobile nodes move. Our top-k query processing methods retrieve the exact answer with small overhead and delay by gradually increasing the search area based on FReT. We demonstrate, through extensive experiments, that FReT and query processing methods function well in terms of small delay and overhead without sacrificing exactness of the query result.

Keywords Top-k query · Replication · Mobile ad hoc networks

1 Introduction

Wireless communication technologies and computer devices have grown remarkably in the past several decades. Mobile devices directly connect with other devices without any

✉ Yuya Sasaki
sasaki@ist.osaka-u.ac.jp

Takahiro Hara
hara@ist.osaka-u.ac.jp

Yoshiharu Ishikawa
ishikawa@i.nagoya-u.ac.jp

[1] Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

[2] Graduate School of Informatics, Nagoya University, Nagoya, Japan

infrastructure by using wireless communication such as bluetooth and Wi-Fi direct. They can easily communicate each other and share data items on demand. These developments bring us a new network concept *mobile ad hoc network (MANET)* [4, 13]. MANETs are opportunistic networks, and composed of autonomous mobile nodes (e.g., people having mobile devices) which can communicate each other without any infrastructures. When a mobile node sends a message to another node which is not within its radio range, other nodes relay the message to successfully reach the destination node. From this feature, MANETs are expected to be developed in many situations such as a disaster. MANETs have some important factors to be widely used such as routing protocols [18] and security [11]. One of the important factors is *query processing*, but it has not been studied well. In this article, we focus on *top-k query* processing in MANETs, which computes the k most relevant data items based on attribute values with regard to a query condition. This fundamental query has been developed in many important applications such as (i) Web and Internet search engines, for word-occurrence and text-based relevance [9]; (ii) wireless sensor networks (WSNs), for detection of statistical outliers [26]; and (iii) peer-to-peer (P2P) networks, for sharing interesting contents [25]. Top-k queries are also available in the field of MANETs such as the following example:

Example 1 We assume a disaster site where the informational infrastructure (e.g., the Internet) is disabled. In this situation, MANETs are helpful because we can share data without specific infrastructures. In a disaster site, numerous rescuers input on-site information on victims and buildings into their devices, and search for seriously-injured victims and badly-damaged buildings. Due to a shortage of supplies or manpower (e.g., the number of ambulances is 10 or the number of groups of rescuers is 50), it is desirable to acquire only important information on the most seriously-injured victims or badly-damaged buildings. For example, rescuers issue a top-k query to retrieve ten victim information with the highest injury levels.

1.1 Motivation and technical overview

Although the top-k query processing on MANETs has promising applications, few studies have investigated the potential of top-k query processing in MANETs [16, 20, 22]. To process top-k queries in MANETs, when a mobile node issues a top-k query (we call a mobile node which issues a top-k query a *query-issuer*), the query-issuer needs to send a query message to all nodes in the entire network for retrieving the exact answer, because the query-issuer does not know which nodes have data items included in the exact answer. However, this procedure is quite inefficient. To avoid this inefficient procedure, *replication* is a promising technique; each node replicates data items retained by other nodes into its storage. Replication is effective in a MANET environment [17] because it has the capability to improve data availability. By using replication, the query-issuer can receive the answer only from few nodes instead of all nodes (i.e., can retrieve the answer in a limited search area). Currently, no study effectively combines top-k query processing with replication in MANETs.

Therefore, we propose a framework that effectively combines a novel top-k query processing method with replication strategy in MANETs to be capable of guaranteeing the exact query results in a limited search area. Mobile nodes replicate data items based on our replication strategy called *FReT* (topology-Free Replication for Top-k query). FReT determines the set of replicas retained by each node (we call it *replica block*) and the ratio that replicas are allocated (we call it *replica ratio*) to efficiently acquire the exact answer.

Moreover, FReT has no maintenance costs due to the movement of the nodes. In our query processing methods, the query-issuer repeatedly sends the query message until it acquires a top-k result, through increasing the TTL (time-to-live) that defines the search area by a hop count. The increase in TTL is based on two complementary approaches; the *expanding ring* and *bundling* methods. The expanding ring method aims at reducing the overhead; on the other hand, the bundling method aims at reducing the delay. These two methods can guarantee the acquisition of perfectly accurate answer with less overhead and delay.

1.2 Contribution and organization

We summarize our contributions as follows:

- Prior work does not effectively acquire an exact answer in environments where mobile nodes hold replicas. Our framework is the first method which combines top-k query processing with replication in MANETs.
- We analyze problems of top-k query processing and replication over a MANET environment.
- We formulate a replication strategy in which the query-issuer can acquire an exact answer in a limited area. The replication strategy has robustness for movement of nodes and dynamical topology change.
- We demonstrate, through extensive simulations taking into account the effect of the physical layer, that our framework functions well in terms of small delay and overhead.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 provides preliminaries. Section 4 presents the proposed method. Section 5 explain an extension of the proposed method. Section 6 summarizes the results obtained in the simulation experiments. Section 7 concludes the paper. This article is an extended version of our previous work [21]. Since the previous work lacks theoretical analysis of our algorithm, we theoretically analyze the performance of our framework and add new insights in this articles. Furthermore, we propose extension techniques and add experimental studies for checking the performance with varying additional parameters.

2 Related work

We describe existing works related to top-k query processing and replication. First, we review some typical replication strategies and cooperative caching protocols in MANETs. Then, we review some typical top-k query processing protocols in a variety of networks such as P2P networks, WSNs, and MANETs.

2.1 Caching and replication in MANETs

First, we review some cooperative caching protocols in MANETs. In cooperative caching, mobile nodes store data items when they receive the data items. On the other hand, in replication, mobile nodes store data items as replicas in advance. Strengths of cooperative caching is that it does not need to distribute data items initially and can store data items according to user's requirements on demand, while the weaknesses is that it cannot achieve an optimal allocation. In [5], the authors proposed a form of cooperative caching called *Hamlet*, aimed at creating diversity within neighboring nodes, so that users would likely find required data items nearby, and thus the network would be less vulnerable to being flooded with query messages. In [8, 23], the authors proposed a caching method for

top-k query processing in MANETs[1]. As each node stores data items based on a summation of a random number and the score of data item that is computed from attribute values with regard to a query condition, data items with high scores are frequently allocated, but diversity of cached data items is maintained by setting an appropriate random number. This method caches undue data items with low scores, and it is difficult to determine an appropriate random number without knowledges of both query conditions and all data items. Thus, this caching method is far from optimal. The authors [23] proposed top-k query processing method for cached data items, but it does not efficiently processes top-k queries because it basically floods query messages into the entire networks.

Second, we review some data replication protocols in MANETs. In [7], the author proposed three algorithms for replica placement, to improve data availability. These algorithms determine the allocation of replicas based on the access frequency, the replicas retained by neighboring nodes, and the network topology. In [12], the authors proposed a form of location-based replication called *location-aided content management architecture* (LACMA for short), which supports searching for required data items in areas in which there is a high probability of being found. In LACMA, data items are allocated to a specific grid based on access frequency, and data items with higher access frequency are specifically allocated to smaller grids. When leaving the current grid, nodes push the data items to bind the replicas to a specific grid. However, if the respective grid contains no nodes, it cannot push the data items.

These protocols except for [8, 23] basically assume that each node requests one data item with one query; however, a rank-based query, such as a top-k query, retrieves several data items with one query. As our proposed protocol aims at acquiring k data items through a small number of nodes that are accessed, without flooding the entire network with queries, it is more difficult to effectively replicate data items, in order to reduce the overhead. As a consequence of our survey, only LACMA [12] easily adapts to top-k query processing within a limited search area because the top-k result may be acquired from specific grids, while the other protocols are hard to process top-k queries within a limited search area because they cannot simply determine the search area. Moreover, since the replication strategies involve maintenance cost due to the movement of nodes, we are aiming for no maintenance cost even if mobile nodes dynamically move.

2.2 Top-k query processing

Numerous methods of top-k processing in a variety of networks have been proposed. A host of top-k query strategies have been proposed with respect to fixed P2P networks and environments that data items are horizontally distributed. In [19], the authors proposed a method in which the query-issuer caches the top-k result, and uses the historical query information. In [24], the authors proposed a method for filtering out unnecessary data items by using *skylines* [2]. However, these strategies consider neither packet collision nor movement of nodes, and thus cannot be directly adopted to MANETs. In addition, typical distributed environments assume that base stations and peers directly communicate each other, but which is impossible in MANETs and WSNs because they are based on wireless communication. In WSNs where data are sent by multi-hop relays to the sink, numerous strategies have been proposed to minimize both the communication cost of data transmission and battery consumption. In [26], an algorithm called *FILA* constructs filters for nodes in order to reduce data sent to the base station (i.e., the sink). In [10], the authors proposed a method which

[1] Although the proposed method in [8] is named a replication method, this method is essentially a caching protocol because each node stores data items when it receives data items.

constructs a top-k filter by utilizing a dominant graph which was discussed in [27] as a data structure for efficient top-k query processing in centralized databases. A WSN environment is more similar to a MANET environment than P2P networks because both WSN and MANET use wireless communication. However, most of these studies in WSNs involved fixed sensor nodes and a single sink.

A few works [16, 20, 22] are existing studies that address the issue of top-k query processing in MANETs. The method proposed in [16] assumed an economic scheme (e.g., virtual currency) with a fundamentally different assumption from that of our study. In the methods proposed in [20, 22], the query-issuer floods a query message into the network, and receives in reply only data items with high scores, the goal being to minimize unnecessary data item replies. Since these methods do not consider data replication, they employ a simple flooding and have difficulty in guaranteeing the acquisition of perfectly accurate query results due to packet losses. In [1], the query-issuer sends a query message to only promising nodes based on a routing table. This work can limit the number of access nodes, however, it does not assume replication, and thus, the query-issuer sends query messages to far nodes (i.e., search area becomes large). Moreover, it needs to maintain routing table with movements of mobile nodes. These top-k query processing protocols in MANETs do not become competitors to our work because they need to flood query messages into the network or send query message to far nodes, which definitely involve a large amount of overhead.

3 Preliminaries

In this section, we explain system model and formulate our problem as preliminaries. Table 1 summarizes the symbols used in the paper.

3.1 System model

The system model is assumed to be a MANET in which mobile nodes retrieve k highest relevant data items (retained by themselves and other mobile nodes) using a top-k query. The query-issuer designates the number of requested data items k and a query condition. The data items are highly relevant when they have high scores that can be computed from a query condition. k is selected from K different values k_i ($i = 1, \ldots, K$) with a probability of $p(k_i)$. We assume that $k_i < k_{i+1}$ and k_K is the maximum number of requested data. For simplicity, all nodes designate k_i with the same probability (i.e., use a given same access model). We assume that each node knows the statistical access rate of k on a given situation.

Table 1 Symbols

Symbol	Meaning
k	Number of requested data items
k_i	K different values of k ($i = 1, ..., K$)
$p(k_i)$	Probability that k_i is designated
M	Number of nodes
m_i	Identifier of node ($i = 1, ..., M$)
D	Number of data items
d_i	Identifier of data ($i = 1, ..., D$)
	i means ranks of data items
ρ	Number of replicas which a node can allocate
R	Communication range

In a real situation, the access rate may not be perfectly known in advance. However, as most existing studies [3, 7] assume that each node knows the access rate for each data item in advance, we also follow it. We assume that the query condition includes a kind of data (e.g., victim information) and its attribute (e.g., injury level). The query condition actually has no restriction since the proposed method is independent of it. In this article, we focus on only single query condition, i.e., all query-issuers issue a specific type of queries. All mobile nodes have opportunities to send top-k queries, i.e., to become query-issuers when they want.

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 $\mathbf{d} = \{d_1, d_2, \cdots, d_D\}$, where D is the total number of data items and $d_i (1 \leq i \leq D)$ is a data identifier. We define the subscript of d as the rank (i.e., d_1 has the highest score and d_k has the k-th highest score). Each data item is initially retained by a specific node, and all data items are assumed to be the same size and not to be updated for simplicity.

We assign a unique *node identifier* to each mobile node in the system. The set of all nodes in the system is denoted by $\mathbf{m} = \{m_1, m_2, \cdots, m_M\}$, where M is the total number of nodes and $m_i (1 \leq i \leq M)$ is a node identifier. Each mobile node moves freely, however, no nodes leave from the network and no additional nodes join. We assume network partitioning does not occur. Each node can allocate ρ data items as replicas. For simplicity, ρ is same for all nodes. Every mobile node has a communication device with communication range R, and recognizes its own location by using a positioning system such as GPS. We do not care about differences in devices (e.g., smart phone and tablet), but we assume that all nodes are the same type of device. Mobile nodes use only a small portion of their storage for replication (i.e., ρ is very small) since mobile nodes issue only one specific type of query.

Note that we employ several assumption for simplicity to solve fundamental problems to process top-k queries and replicate data items in MANETs. However, our proposed approach works well without some assumptions. In Section 5, we discuss extension of our proposed approach to adapt to general situations.

3.2 Problem formulations

Problems of top-k query processing, replication, and MANETs are intricately intertwined. In this section, we analyze the problems.

3.2.1 Problems for top-k query processing in MANETs

The query-issuer transmits a query message with the given query condition and k over the entire network, in order to acquire k data items with the highest scores.

Definition 1 Given the set of data $\mathbf{d}$, the set of mobile nodes $\mathbf{m}$, the number of data items k, and a query condition, a top-k query problem is to acquire k data items with the highest scores (top-k result), with the least overhead and delay.

In top-k queries, the query-issuer does not designate the data identifier, so it cannot judge whether the received data items are perfectly accurate or not even if it receives k data items. More specifically, because of packet losses, some data items within the k-th rank may be missing and others outside the rank can be included in the result. If each node knows the data identifier of the top-k result, it guarantees the exact answer and also effectively allocate replicas. Thus, in our framework, first a node collects the top-k result and distributes

information on the top-k result to all nodes in the MANET, and each node determines replicas based on the information.

Our goal is to develop efficient algorithms for top-k query processing in MANETs. We consider several performance measures in designing our algorithms: (1) accuracy of query result (higher is better), (2) communication overhead (smaller is better), and (3) delay to acquire the top-k result after issuing a query (smaller is better). The accuracy is the most important measure because if the accuracy is very low, small overhead and delay have no meaning, so a better goal is to keep the perfect accuracy and decrease the overhead and delay as much as possible. Moreover, there is a trade-off between the overhead and delay, and thus we should consider together. Both communication overhead and delay decrease when the number of nodes that must be accessed in order to acquire the result decreases. Because, if the number of nodes that must be accessed is small, the query-issuer can acquire the result from only nearby mobile nodes. Therefore, in this paper, the optimal data allocation reduces the average number $f(\mathbf{k})$ of nodes that must be accessed to acquire the result over the entire system.

Definition 2 The optimal data allocation for top-k queries over MANETs achieves the smallest $f(\mathbf{k})$, which is calculated based on the following equation:

$$f(\mathbf{k}) = \sum_{i=1}^{K} p(k_i) \cdot f(k_i), \tag{1}$$

where $f(k_i)$ is the average number of nodes that must be accessed when the query-issuer designates k_i.

3.2.2 Replication problems for top-k queries

Replication strategies for single data item access are not effective for top-k queries. This is because the access ratio for data items in a top-k query is strongly biased (e.g., d_1 is always accessed). For example, *Square root* replication [3] has been proposed as possible means of determining the optimal number of replicas for single data item access in unstructured P2P networks. Square root replication establishes the optimal ratio of replication for d_i based on the following equation:

$$r_i = \frac{\sqrt{q_i}}{\sum_{j=1}^{k_K} \sqrt{q_j}} \text{ and } l \leq r_i \leq u \tag{2}$$

where q_i is the access rate for d_i, k_K is the maximum k, and l ($\geq \frac{1}{M \cdot \rho}$) and u ($\leq \frac{1}{\rho}$) denote the minimum and maximum r_i (every data item must be replicated by at least one node, and every node must not allocate more than one replica of the same data item).

In top-k queries, data items with higher ranks are more frequently accessed than those with lower ranks. Therefore, the ratio of replication for d_i is higher than that for d_{i+1}.

$$r_i \geq r_{i+1}. \tag{3}$$

However, if we use a simple strategy which the ratio of replication is determined based only on the access rate, the data items with high ranks are allocated too much, resulting in a lack of diversity of replicas. In that case, if large k is specified and the query-issuer needs to acquire data items with low ranks, both overhead and delay may increase. Moreover, a replication strategy for single data item access does not take into account the fact that in top-k queries, some data items are dependently accessed, i.e., data items with similar ranks are often accessed by the same query.

3.2.3 Replication problems for MANETs

In MANETs, the overhead involved in query processing is significantly lessened when the necessary data items are replicated near the query-issuer. Therefore, location-based replication has been proposed in MANETs and WSNs. Here, the overhead required for query processing is calculated based on the following equation:

$$cost = \sum_{x \in \mathbf{m}} \sum_{i=1}^{K} p(k_i) \sum_{j=1}^{k_i} hop(x, d_j) \tag{4}$$

where $hop(x, d_j)$ denotes the number of hop counts between the query-issuer (x) and the node that retains a replica of d_j. Optimal replication achieves the minimum $cost$, but it is known to be an NP-hard problem. Moreover, nodes move freely in MANETs, and thus the optimal replication changes dynamically. In addition, since the number of neighboring nodes changes dynamically, the search area for acquiring necessary data items cannot be known in advance. It may be possible to reallocate replicas every time mobile nodes move, but it involves significant overhead instead of query processing. Therefore, a strategy is needed which involves no maintenance overhead even if nodes move, and it determines the relevant search area on the fly. Here, it should be noted that perfect optimal replication is impossible in MANETs unless all nodes know all nodes' mobility patterns, query timings and the network topology.

4 Framework for processing top-k queries with replication

In this section, we first describe design policy of our framework. Then, we present the replication strategy, the initial collection and distribution methods, and the top-k query processing method.

4.1 Design policy

We design our replication strategy and query processing methods for top-k query based on the following policy. To achieve small $f(\mathbf{k})$ in Eq. (1), we aim at reducing the number of hop counts for retrieving the result. This is because if the number of hop counts is small, both $f(\mathbf{k})$ and the cost in Eq. (4) become small.

A naive approach is that we uniformly allocate data by taking into account the network topology and locations of mobile nodes. However, since the network topology of MANETs dynamically and frequently changes, we need to reallocate when mobile nodes move. Since this reallocation takes a large overhead, we need an approach without reallocation. For avoiding reallocation, we do not use the topology information and locations of mobile nodes.

We allocate data based on the probabilities of data access to keep a variety of allocated data. If we simply use the probabilities of data access such as Eq. (2), allocated replicas lack the diversity. Thus, to determine the ratio of allocated data, we do not use the access rate for data items, but we use the probability of the number of requested data k. In addition, in MANETs, the mobile nodes freely move, and then the neighbor nodes frequently change. While, the number of neighbor nodes is relatively constant compared with the network topology. As a result, we calculate the ratio of allocated data by using based on the probabilities of k and the number of neighbor nodes.

Although we achieve efficient data allocations, mobile nodes do not know the locations of mobile nodes that have their necessary data items. Therefore, we need to decide a search

range represented by the number of hop counts. If the number of hop counts is small, we may need to iteratively send messages until receiving the result of a top-k query. On the other hand, if the number of hop counts is large, data may be replied back from far mobile nodes which causes the large overhead. Above two cases have a trade-off between the overhead and the delay. Hence, we develop two message processing methods; expanding ring method and bundling method, which are inspired by our previous work [22]. Expanding ring method gradually increases the number of hop counts one by one until receiving the result. The expanding ring method reduces an overhead for query processing. On the other hand, the bundling method first collects data from one hop nodes, and then it increases the number of hop counts by using the information of the number of nodes around itself. The bundling method reduces a delay for query processing. Of course, both method can reduce the number of hop counts because of our replication strategy. By the combinations between our replication strategy and message processing methods, we can efficiently process the top-k query.

4.2 FReT: replication strategy

In our replication strategy FReT, mobile nodes allocate *replica block* which is the set of data with consecutive ranks. The replica block is allocated based on *replica ratio* which is a probability that replica block is allocated. We first explain the replica block and the replica ratio, and then we describe an algorithm of FReT.

4.2.1 Replica block

In top-k queries, data with similar ranks are accessed together with a high probability for a given k. Therefore, each node allocates ρ data with consecutive ranks (i.e., d_1 through d_ρ, $d_{\rho+1}$ through $d_{2\cdot\rho}$, $\cdots$, and $d_{(\lceil \frac{k_K}{\rho} \rceil - 1)\cdot\rho+1}$ through d_{k_K}). We call a set of ρ data with consecutive ranks *replica block* $\mathbf{b} \in b_i$ $(1 \leq i \leq \lceil \frac{k_K}{\rho} \rceil)$. Each mobile nodes allocate replica blocks.

4.2.2 Replica ratio

Each replica block is allocated based on the *replica ratio* $p(\mathbf{b})$ which are the probabilities that the replica blocks are allocated (e.g., $p(b_1)$ is the replica ratio for b_1). In our replication strategy, the replica ratio is calculated for efficiently processing top-k queries. The query-issuer can acquire the result from a small number of nodes if we efficiently allocate the replica block.

The number of nodes that are accessed depends on both of the number of neighbors of each node (i.e., nodes within its radio range) and the number of hop counts from a given node. When the numbers of hop counts and/or neighbors are large, the number of nodes that are accessed is large. Given the number of hop count h, we calculate the expected number M_h of nodes that are accessed based on the following equation:

$$M_h = 1 + M_a \times \sum_{j=1}^{h} j \tag{5}$$

where M_a denotes the average number of neighbors for all nodes (M_a is able to be aware after an initial collection described in next section). In this equation, we assume that nodes uniformly exist, and all nodes have the same number of neighbors (M_a). If the number of

neighbors is extremely large, this estimation is not very precise, but we do not assume that the number of neighbors is not much large.

Next, we define $p(k_i, M_h)$ as the probability that the top-k_i result is acquired by accessing M_h nodes:

$$\begin{aligned} p(k_i, M_h) \;=\; & 1 - (\overline{p(b_1)} + \cdots + \overline{p(b_{\lceil \frac{k_i}{\rho} \rceil})}) \\ & + (\overline{p(b_1) + p(b_2)} + \cdots + \overline{p(b_{\lceil \frac{k_i}{\rho} \rceil - 1}) + p(b_{\lceil \frac{k_i}{\rho} \rceil})}) \\ & + \cdots \\ & + (-1)^{\lceil \frac{k_i}{\rho} \rceil} \cdot \overline{p(b_1) + \cdots + p(b_{\lceil \frac{k_i}{\rho} \rceil})} \end{aligned} \tag{6}$$

where $\overline{x}$ denotes $(1 - x)^{M_h}$. This equation calculates the probability of a complementary event that a given node cannot access replica blocks: $p(b_1)$ to $p(b_{\lceil \frac{k_i}{\rho} \rceil})$. If M_h is less than $\lceil \frac{k_i}{\rho} \rceil$, $p(k_i, M_h)$ becomes 0. $p(k_i, M_h)$ increases as M_h (i.e., h) increases, because the probability that the node cannot access necessary data decreases. On the other hand, a large M_h indicates a large overhead. Thus, smaller M_h which achieves larger $p(k_i, M_h)$ is better. We determine the optimal h for processing top-k_i queries so that we achieve the smallest $\frac{M_h}{p(k_i, M_h)}$. $p(k_i, M_h)$ increases when data d_1 through d_{k_i} are frequently allocated (i.e., $p(b_1)$ to $p(b_{\lceil \frac{k_i}{\rho} \rceil})$). However, if we prioritize only small k, the diversity of allocated data decreases because data items with high ranks significantly increases.

Therefore, we should calculate M_h for every k_i, M_{h_i}, to totally achieve a small number of nodes that must be accessed in the entire MANET. For this aim, we calculate $f(\mathbf{k})$ based on the following equation:

$$f(\mathbf{k}) = \sum_{i=1}^{K} \left(p(k_i) \cdot \frac{M_{h_i}}{p(k_i, M_{h_i})} \right). \tag{7}$$

If $f(\mathbf{k})$ is minimized, the query-issuer can acquire the top-k result by searching a small number of nodes with a high probability. Therefore, in our replication strategy, the replica ratio is established so as to achieve minimum $f(\mathbf{k})$.

4.2.3 Algorithm of FReT

To calculate $f(\mathbf{k})$, we recursively calculate M_h at the various $p(\mathbf{b})$. Algorithm 1 shows the pseudo-code to establish $f(\mathbf{k})$. In this algorithm, δ denotes a parameter value which shreds $p(\mathbf{b})$ evenly. A smaller δ can calculate more precise replica ratio, but increases the computation cost. We can control δ based on a computational power of mobile nodes. The algorithm iteratively computes $f(k_i)_h$ for each k_i and h until $f(k_i)_h \geq f(k_i)_{h-1}$ (lines 4 – 7). Since $f(k_i)_h$ first decreases and then increase, $f(k_i)_{h-1}$ is the minimum value when it satisfies that $f(k_i)_h \geq f(k_i)_{h-1}$. It keeps the minimum $f(k_i)$ for each i, and then $f(\mathbf{k})$ is calculated by summing all $f(k_i)$ for a given $p(\mathbf{b})$ (lines 8 – 13). We calculate all patterns of $p(\mathbf{b})$ (recall that $p(b_i) \geq p(b_{i+1})$) (line 17). To reduce the computation cost, the b_i with a same access ratio are calculated at once. Finally, after all patterns of $p(\mathbf{b})$ are checked, it returns $p(\mathbf{b})$ with minimum $f(\mathbf{k})$.

Algorithm 1 FReT algorithm.

Require: ρ, M_a, $p(k_i)$, and δ
Ensure: $p(\mathbf{b})$ with the minimum $f(\mathbf{k})$
1: $b_{num} = \lceil \frac{k_K}{\rho} \rceil$
2: $b_i = \frac{1.0}{b_{num}}$ for $i = 1, \cdots, b_{num}$
3: **while not** all patterns of $p(\mathbf{b})$ are done **do**
4: **for** $i = 1, \cdots, K$ **do**
5: **for** $h = 1, \cdots$ **do**
6: $M_{h_i} = 1 + M_a \cdot \sum_{j=1}^{h} j$
7: $f(k_i)_h = \frac{M_{h_i}}{p(k_i, M_{h_i})}$
8: **if** $h \neq 1$ **and** $f(k_i)_h \geq f(k_i)_{h-1}$ **then**
9: $f(k_i) = f(k_i)_{h-1}$ and **break**
10: **end if**
11: **end for**
12: **end for**
13: $f(\mathbf{k}) = \sum_{i=1}^{K} p(k_i) \cdot f(k_i)$
14: **if** $minF > f(\mathbf{k})$ **then**
15: $minF = f(\mathbf{k})$, and $result \leftarrow p(\mathbf{b})$
16: **end if**
17: Calculate next $p(\mathbf{b})$ based on δ
18: **end while**

The time complexity of FReT is as follows.

Theorem 1 *Given the maximum hop h_m, ρ, $\mathbf{k}$, and δ, the time complexity of FReT is*

$$O\left(K \cdot h_m \cdot \left(\lceil \frac{k_K}{\rho} \rceil \cdot \frac{1}{\delta}\right)^{\lceil \frac{k_K}{\rho} \rceil}\right). \tag{8}$$

Proof The algorithm of FReT iteratively computes $f(k_i)_h$ for the number of requested data items, the number of hop counts, and patterns of $p(\mathbf{b})$. The patterns of $p(\mathbf{b})$ are permutations of the probabilities with constraints such as $p(b_i) \geq p(b_{i+1})$ and sum of $p(\mathbf{b})$ is one. The number of patterns of $p(\mathbf{b})$ is $O\left(\lceil \frac{k_K}{\rho} \rceil \cdot \frac{1}{\delta}\right)^{\lceil \frac{k_K}{\rho} \rceil}$. Therefore, the computation cost of FReT becomes Eq. (8). □

Here, the maximum hop is computed from the size of area and communication range practically.

Example 2 Let ρ be 5, M_a be 10, $k_i (i = 1, \cdots, 4)$ be $\{25, 50, 75, 100\}$, $p(k_i)$ be $\{0.25, 0.25, 0.25, 0.25\}$ and δ be 0.001. The number of replica block is 20 ($= \lceil \frac{k_K}{\rho} \rceil = \lceil \frac{100}{5} \rceil$). Here, since b_1 through b_5 (denoted by b_{1-5} for short) are accessed with a same probability, they have same replica ratio. Similarly, b_{6-10}, b_{11-15}, and b_{16-20} have the same replica ratio, respectively. Hence, we calculate the four kinds of replica ratios to reduce a computation cost. Finally, $p(b_{1-5})$, $p(b_{6-10})$, $p(b_{11-15})$, and $p(b_{16-20})$ are 0.07, 0.052, 0.042, and 0.036, respectively.

FReT is a high robust replication strategy because the replica ratio does not sensitively change unless the network topology significantly changes. FReT establishes the replica ratios only based on the average number of neighbors for all nodes and access rates, which are relatively constant. Since locations of nodes and the network topology frequently change in

MANETs, we do not use locations and topological information of nodes for our replication strategy. Thus, our replication strategy achieves high robustness for moving nodes.

4.3 Initial collection and dissemination

To optimally allocate replicas, each node should know the network information in the entire network and the data identifiers of the top-k result. Therefore, our framework collects and disseminates the network information and the identifiers of the top-k result. We call the processes for collecting and distributing data *initial collection* and *initial dissemination*, respectively. In these process, the node which is the first query-issuer, m_c, becomes a *coordinator*. In the initial collection, m_c transmits an *initial message* to every node. Then, every node replies back their location and data items with high scores that are possible to be the top-k result. To reduce the overhead for initial collection as possible, we employ a combination of the top-k query processing method in [6] and the location-based flooding method in [15].[2] After m_c completes the initial collection, it knows the top-k result and calculates the average number of neighboring nodes. To calculate the number of neighboring nodes, m_c simply compares the distance between all pairs of two node, and if the distance is within R, the two nodes are neighbor nodes each other. In the initial dissemination, m_c disseminates the top-k result and the average number of neighboring nodes by an existing flooding technique. Receivers store the data identifiers and scores of top-k result and the average number of neighboring nodes. Each node calculates the replica ratio based on FReT, and allocates replicas. Receivers are aware of the top-k result, but since their storage has a limit, they store only the data identifiers and scores instead of whole data items, except for their allocated replicas.

Here, we discuss the costs of initial collection and dissemination. The cost of initial collection equals to the cost of top-k query processing to retrieve the k_K data items without replication. This is because the initial collection is equivalent to process the top-k_K query for acquiring the data items with the k_K highest scores. The cost of initial dissemination equals to the flooding for sending the k_K data items. In the naive initial and dissemination methods, the cost is $O(k_K \cdot M)$ because each node sends the k_K data items. As mentioned above, our initial and dissemination methods reduces the cost by using existing top-k query processing and the location-based flooding, respectively. The effects of our methods are shown in Section 6.3.3.

4.3.1 Algorithm of initial collection

Algorithm 2 is the pseudo-code for initial collection. In Algorithm 2, the coordinator m_c sends the initial message to its neighbors (line 1). If node m_r receives the initial message, it stores the information on the initial message, and then sets a query timer to determine when it sends its initial message (lines 3 – 6). Before expiring the time, m_r overhears the reply messages that other nodes send and stores the replied data items for efficiently collecting the top-k result (line 7). If m_r expires its query timer, it calculates communication ranges of sender nodes. If the communication range of m_r is not covered by other sender, it sends the initial message (lines 9 – 13). If the communication is covered, it sends a reply message to its parent (i.e., the node that send the initial message to m_r) (line 15). Then, if m_r expires a reply timer for reply message or receives replies from all of its children, it sends its reply

[2]The reason that we select the top-k query processing method proposed in [6] is that this method does not require any information in advance of query processing

message to its parent (lines 18 – 20). If m_c receives replies from all its children, it starts the initial dissemination phase (lines 21 –23). On the other hand, if m_c waits a predetermined maximum wait time, it resents the initial message (lines 24 – 26).

Algorithm 2 Initial collection.

Require: k_K and a query condition
Ensure: Data items with k_K highest scores and positions of nodes
1: m_c broadcasts IM
2: **if** Node, m_r receives IM **then**
3: **if** Receives first **then**
4: Stores the information on IM
5: Sets a query timer
6: **end if**
7: Updates replied data items and neighbor nodes
8: **end if**
9: **if** m_r expires its query timer **then**
10: Calculates communication range of neighbor nodes
11: **if** Communication range of m_r is not covered **then**
12: Broadcasts IM
13: Sets a reply timer
14: **else**
15: Sends a reply message to its parent
16: **end if**
17: **end if**
18: **if** m_r expires its reply timer and has no children **or** receives replies from its all children **then**
19: Sends a reply message to its parent
20: **end if**
21: **if** m_c receives replies from all its children **then**
22: Starts disseminating the collected information by using the location-based flooding method
23: **end if**
24: **if** $max\ wait$ has passed from sending IM by m_c **then**
25: m_c broadcasts IM again
26: **end if**

Example 3 Recall the example in Section 4.2. Each node receives M_a (= 10) and the data items with top 100th scores. It calculates the replica ratio, and then selects and replicates one of the replica blocks. For example, to select a replica block, nodes generate a random value whose range is [0, 1], and then replicate data items according to the selected replica block. Note that nodes can select a replica block to surely follow the replica ratio (e.g., by using the identifiers of mobile nodes) without randomness because we assume a collaboration work such as a rescue operation.

4.4 Top-k query processing

In this section, we describe how to process top-k queries. We first explain contents of messages and a method for message processing. Then, we explain two methods for increasing the TTL; the *expanding ring* and *bundling* methods.

4.4.1 Query and reply messages

Our message processing method has two phase. The query-issuer initially sends a *first query* in the first phase, and then repeatedly sends a *reiterate query* until the result is acquired in the second phase. We explain the contents of the first query message, the reiterate query message, and the reply message, respectively. The first query message, FQ, includes the identifier of the query-issuer, the identifier of the query, a query condition, k, the list of data retained by the query-issuer ($list_q$). The reiterate query message, RQ, on the other hand, includes the identifier of the query-issuer, the identifier of the query, the identifiers of the demanded data, the identifier of the sender of the message, the position of the sender of the message, and the hop count T. Both reply messages include the identifier of the query-issuer, the identifier of the query, the identifier of the sender of the message, the identifier of the node that previously sends the message before the sender, and the list of reply data ($\mathbf{d}_r$).

There are three main differences between the first query and the reiterate query. First, the former designates k and a query condition, but the latter designates demanded data. This is because the combined size of k and the query condition is less than that of all the identifiers of demanded data, but the query-issuer cannot designate only demanded data in the reiterate query by designating k and the query condition. In the first query, we use $list_q$ to reduce the number of such duplicate reply data. This is because duplicate reply data may be sent in reply messages from nodes though the query-issuer has those replica block. Second, the latter uses location-based flooding, but the former does not. This is because the reiterate query in which the hop count is large has more chance of employing location-based flooding, but the first query in which the hop count is one does not. Thus, in the case of the first query, the location information unnecessarily increases the message size. Finally, the latter sets the hop count to expand the search range, but the former does not set (i.e., the hop count is always one). It has two reasons; to acquire the exact result from nearby nodes, and to know the current number of neighboring nodes as soon as possible.

We here explain that the reiterate query is effective for reducing the overhead. Let suppose that the size of identifier of mobile nodes, identifier of data items, the identifier of the query, a query condition, k are the same. We mentioned above that the first query includes k, a query condition, and $list_q$, while the reiterate query includes the identifiers of the demanded data. Thus, when query-issuers have necessary $\frac{k}{2} - 1$ data items for top-k queries, the reiterate query is more effective than the first query. Additionally, the reiterate query includes the location for using the location-based flooding. Let suppose that the size of location increases α percentages of reiterate queries. If the location-based flooding reduces α percentage, it works well to reduce the overhead.

4.4.2 Algorithm of query processing

The processing methods of the first query and the reiterate query are shown in Algorithms 3 and 4, respectively. In both algorithms, m_q and m_r denote the query-issuer and a node that receives a query message, respectively. In Algorithm 3, if m_q already has the data items with k highest scores, the query is terminated without sending any messages (lines 1 – 2). Otherwise, m_q sends the first query to its neighbors (lines 3 – 4). If m_r receives the first query, it stores the information on the first query such as k, the query condition, and $list_q$, and then sets a timer to determine when it sends its reply message (lines 6 – 9). Before expiring the time, m_r overhears the reply messages that other nodes send and

stores the replied data items $\mathbf{d}_r$ for avoiding sending the same data items (lines 10 – 12). After expiring the timer, m_r sends the reply messages that include its replicas that are not included $list_q$ and are not overheard to m_q (lines 13 – 18). Here, even if it has no replied data items, it sends the reply message to know the number of neighbors. If the query-issuer m_q receives all the data items with k highest scores, the query is terminated (lines 19 – 21). If m_q waits the maximum time for reply timer, it stores the number of nodes that sends the reply messages for computing the TTL of the bundling method (lines 22 – 23). Then, it sends reiterate queries (lines 24).

In Algorithm 4, m_q sends the reiterate query (line 1). If m_r receives the reiterate query first time, it stores the information on the reiterate query and decrease TTL in the received reiterate query (lines 2 –5). If it has the demanded data items, it sends a reply message to its parent and updates the demanded data items (lines 6 – 10). Then, if TTL is larger than zero and the demanded data items is not null, it sends the reiterate query and sets a timer (lines 11 –14). m_r also updates its neighbor nodes regardless that it receives the reiterate query first or not (line 16). If m_r expires the timer, it calculates communication ranges of sender nodes. If the communication range of m_r is not covered by other sender, it sends the reiterate query (lines 18 – 23). Then, if m_r receives a reply message, it updates the replied data items for avoiding sending the data items that are already sent to the parent or sent by other nodes (lines 24 – 27). If the replied data items are not null, it sends the reply message to its parent (lines 28 – 29). m_r also overhears reply messages that are not sent to m_r to know the data items that are sent by other nodes (line 31 – 32). If m_q receives the data items with the k highest scores, the query is terminated (lines 34 – 36). Otherwise, it resends the reiterate query with increasing TTL after waiting the time computed from the TTL (line 37 – 40.)

Algorithm 3 First query processing.

Require: k and a query condition
Ensure: data items with the k highest scores
1: **if** m_q has k data items with the highest scores as its replicas **then**
2: Query is terminated
3: **else**
4: Query issuer m_q broadcasts FQ
5: **end if**
6: **if** Node m_r receives FQ **then**
7: Stores the information on FQ
8: Sets a reply timer as a random value
9: **end if**
10: **if** m_r overhears a reply message **then**
11: Stores $\mathbf{d}_r$ of the reply message
12: **end if**
13: **if** m_r expires its reply timer **then**
14: **for** Replicas, d_i held by m_r that are not included $list_q$ and are not overheard **do**
15: $\mathbf{d}_r \leftarrow \mathbf{d}_r \cup d_i$
16: **end for**
17: Sends a reply message to m_q
18: **end if**
19: **if** m_q acquires the top-k result **then**
20: Query is terminated
21: **end if**
22: **if** m_q waits the maximum time for reply timer **then**
23: Stores the number of nodes replied for the bundling method
24: Go to Algorithm 3
25: **end if**

Algorithm 4 Reiterate query processing.

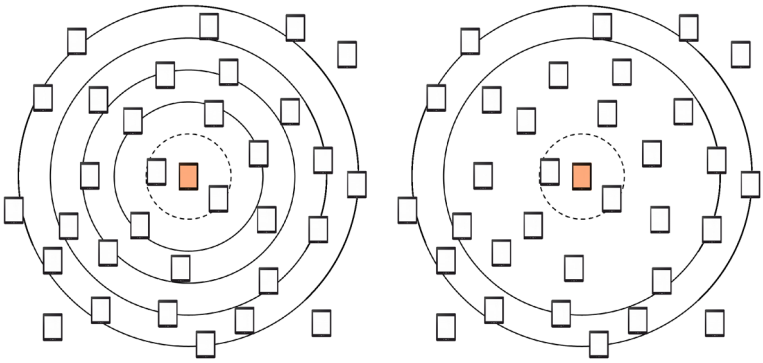
Require: Identifiers of demanded data items
Ensure: Demanded data items
m_q broadcasts RQ
if Node, m_r receives RQ **then**
  if Receives first **then**
    Stores the information on RQ
    Decreases TTL by 1
    if Has demanded data items **then**
      d $\leftarrow$ the demanded data items retained by m_r
      Sends a reply message to its parent
      Demanded data items $\leftarrow$ demanded data items - **d**
    end if
    if $TTL > 0$ **and** demanded data items $\neq \phi$ **then**
      Updates RQ
      Sets a query timer as a random value
    end if
  end if
  Updates neighbor nodes
end if
if m_r expires its query timer **then**
  Calculates communication range of sender nodes based on location-based flooding [23]
  if Communication range of m_r is not covered **then**
    Broadcasts RQ
  end if
end if
if m_r receives a reply message **then**
  for Reply data items d_i are not sent to parent node, and are not overheard **do**
    $\mathbf{d}_r \leftarrow \mathbf{d}_r \cup d_i$
  end for
  if $\mathbf{d} \neq \phi$ **then**
    Sends a reply message to its parent
  end if
else if m_r overhears a reply message **then**
  Stores $\mathbf{d}_r$ of the reply message
end if
if m_q acquires the top-k result **then**
  Query is terminated
end if
if m_q expires a timer determined based on TTL **then**
  Updates RQ
  m_q broadcasts RQ again
end if

4.4.3 Methods for increasing TTL

We propose the expanding ring method and the bundling method. The difference between the two methods lies in the means of setting the TTL.

The expanding ring method was proposed for unstructured P2P networks in [14]. It increases the TTL gradually to minimize the number of nodes that are accessed. In our version of the expanding ring method, the TTL of every first query is set as 1, and the TTL of the reiterate query is set as the previous TTL plus 1. The simple modification makes it possible to minimize the number of reply data items.

The bundling method largely increases the TTL at once. In the method, the query-issuer sets the TTL based on the current number of neighboring nodes M_c. The TTL of the first query is set as 1 because the query-issuer does not know its M_c in advance. After the first



(a) Expanding ring method (b) Bundling method

Fig. 1 Differences in TTL between the expanding ring and bundling methods

query, since the query-issuer now knows its M_c, it increases the TTL of the first reiterate query at once. The TTL is determined to fulfill the following equation when the query-issuer designates k_i as k:

$$M_c \times \sum_{j=1}^{TTL} j > \frac{M_{h_i}}{p(k_i, M_{h_i})} \tag{9}$$

where M_{h_i} and $p(k_i, M_{h_i})$ are the expected number of nodes that must be accessed in order to acquire the data items with k_i highest scores and the probability that the top-k result is acquired by searching M_{h_i} nodes in Eq. (7), respectively. This equation calculates the TTL (≥ 2) that the query-issuer acquires the top-k result with high probability and small number of nodes that are accessed, based on the current number of neighbors. If the query-issuer cannot acquire the top-k result using this TTL, then it increases TTL by 1. This method can increase the TTL at once, potentially lessening the delay.

Figure 1 shows an image of increasing TTL in the two methods. In Fig. 1, orange and white devices denote the query-issuer and receivers, respectively. Dashed and solid circles denote the search areas of the first and reiterate queries, respectively. In Fig. 1a, the expanding ring method increases the TTL by one until the query-issuer receives the top-k result. Thus, the search area in the expanding ring method expands gradually. In Fig. 1b, the bundling method has a large TTL of the first reiterate query because the number of neighbors is small. After increasing the TTL of the first reiterate query, the bundling method increases the TTL by one until the query-issuer receives the top-k result as well as the expanding ring method.

The difference between these methods is the TTL of the first reiterate query. In the expanding ring method, the TTL of the first reiterate query is always two even if the network is very sparse. On the other hand, the bundling method changes the TTL of the first reiterate query depending on the number of neighboring nodes of the query-issuer, that is the density of nodes. The density of nodes affects the number of hop counts for retrieving the top-k result. If the network is dense, the number of hop counts becomes small. On the other hand, if the network is sparse, the number of hop counts becomes large. We can compute the probability that the query-issuer acquires the top-k result by Eq. (6). The number of accessed nodes M_h in Eq. (6) is computed from the current number of neighbors and the specified TTL. When the density of nodes is small, the number of reiterate queries increases. Thus, the delay in the expanding ring method may increase while it achieves a

small overhead. Meanwhile, the bundling method can effectively reduce the delay compared with the expanding ring method. However, due to an excessive increase of TTL, the number of reply data items is basically greater than that in the expanding ring method. The two methods have a tradeoff between delay and overhead.

5 Extension techniques

In this section, we explain two extension techniques for our framework; location-based allocation and data update.

5.1 Location-based allocation

Our framework does not use location information for avoiding maintenance costs due to movement of nodes. It is very efficient because nodes basically move frequently. However, if nodes do not move frequently, the location-based allocation is efficient. Thus, we propose an allocation technique based on location information.

When we use location information for allocating replicas, a grid-based replication is a basic approach [12]. Thus, we propose a grid-based allocation whose grids are hexagon; a regular hexagon fills the most fraction of its circumscribed circles among polygons that can tile the plane (see Fig. 2). First, let us describe the deployment of hexagonal grids. The grids, $g(i, j)$, cover the area and are deployed next to neighboring inscribed hexagons. The i and j in the grid identifier, $g(i, j)$, correspond to the column and the row numbers, respectively.

Each node determines what data items are allocated based on the grid that it belongs to. We allocate the replicas so that each node retrieves the k_α highest scores from its grid and k_K highest scores from its and neighboring grids. We set k_α to the number that satisfies $\sum_{i=1}^{\alpha} p(k_i) \geq 0.5$. Thus, each grid should include $k_\alpha + \frac{k_K - k_\alpha}{7}/\rho$ nodes. This is because our approach searches for k_K data items with the highest scores from nodes in seven grids. The diameter of grid is the minimum $i \cdot R(i = 1, 2, \cdots)$ that all grids include $(k_\alpha + \lceil \frac{k_K - k_\alpha}{7} \rceil)/\rho$ nodes. Since R is the communication rage of nodes, we increase the diameter based on R. We allocate data items to grids according to the replica ratio. In more concretely, we determine allocated replicas from identifier of grids $g(i, j)$. Here, since the grid does not

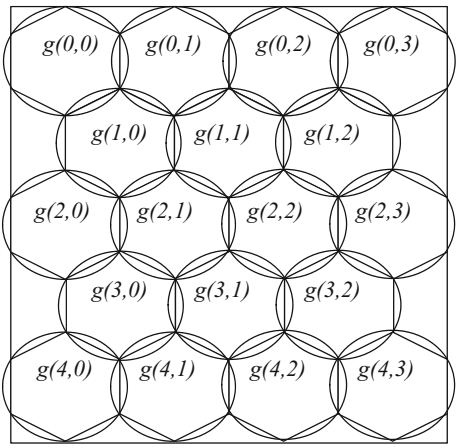


Fig. 2 Grid deployment

typically tile the whole area and some nodes do not belong to any grids, such nodes allocate replicas following the replica ratio. We can efficiently search for top-k data items by the grid-based initial allocation if nodes do not move.

Reallocation is needed when a node leaves its own grid. A node notifies nodes in the source grid that it is leaving and nodes in the destination grid that it is participating, respectively. The node sends a *moving message* to neighboring nodes when it leaves its own grid. The moving message is attached to replicas that should be reallocated, data item identifiers whose replicas it has, the available space information on this nodes (the time that it deletes the replicas), and grid identifiers of the source and destination grids. A node that received the message in the source grid allocates the received replicas if it has available space. A node that received the message in the destination grid removes data items attached to the message to reduce the message size and it attaches the information of its own grid to notify the information to moving node after updating the information of the grid. Both nodes in the source and destination grids transmit messages to their neighboring nodes when the grid size is more than $2 \cdot R$. Each node can know the information of its own grid with this procedure.

We here discuss the tradeoff between methods with and without using locations. Let p_m be a probability that nodes move out from their areas and o_m be an maintenance cost for the movement. In addition, let o_q be the difference of overheads for query processing between method with and without using locations. $p_m \cdot o_m$ can be considered as the average maintenance cost for the movements. If $p_m \cdot o_m$ is smaller than o_q, the location-based approach works well. Otherwise, the location-based approach suffers a large maintenance cost even if it can reduce the overhead for query processing.

5.2 Data update

When data items are updated, the top-k result may change. In this case, it is difficult to guarantee the perfect accuracy of the query result. Our framework easily can adapt to the data update. We have three procedures for handling data update; (1) initial collection and dissemination, (2) message processing for data generation, and (3) message processing for data deletion.

Initial collection and dissemination: It takes a large cost to find new top-k data when data items in the top-k result are deleted. For avoiding searching for new top-k data from scratch, in the initial collection, the coordinator collects the data items with ($k_K + margin$) highest scores. *margin* is a positive integer which is computed from probabilities that data items in the top-k results are deleted. The probability is computed from k_K the ratio of data update. In the initial dissemination, the coordinator disseminates the data items in the same way, and then each node replicates these data items.

Message processing for data deletion: When a node deletes data items with $k_K + margin$ highest scores, the node must notify all nodes of the deletion because the top-k result changes. Thus, it sends a *deletion message* in the network to notice the deletion of the data items. The deletion message contains the identifier of the deleted data items. All nodes that receive the message changes the scores of top-k result and data items in the margin. If they have the deleted data items as replicas, it deletes the replicas. Even if some data items in the top-k result are deleted, we do not need to search for new data items, because nodes allocate additional data items with $k_K + margin$ highest scores. Thus, we can guarantee the top-k result without reallocation.

Message processing for data generation: When a node generates new data items with $k_K + margin$ highest scores, the top-k result changes. Thus, it sends a *generation message*

in the network to notice the generation of the data items. The generation message contains the new data items. All nodes that receive the message changes the scores of top-k result and data items in the margin. If nodes have replicas that become out of data items with $k_K + margin$ highest scores, the nodes having those replicas replace them with new ones. If data items are deleted before generation, nodes that have spaces to allocate replicas replicate the new data items. This approach does not need to exchange extra messages and data items among nodes.

6 Performance evaluation

In this section, we summarize the results of experiments evaluating performance of our framework.

6.1 Experimental model

For the experiments, we used the network simulator, QualNet5.2, which takes into account the effect of physical layer (i.e., packet losses and delays occur due to radio interference).[3] Mobile nodes are present in an area of 1000 meters $\times$ 1000 meters. The initial position of each node is determined randomly. The number of mobile nodes is M (default setting is 500). The nodes move according to the random-walk model, with a random velocity range of 0.5 to v (default setting is 1) meters/second (when v is 0, nodes are stationary). Each mobile node transmits messages (and data items) using an IEEE 802.11b device, with data transmission rate of 11 Mbps. The transmission power of each mobile node is set such that the radio communication range R is roughly 100 meters.

Each mobile node retains 100 data. The size of a data is s (default setting is 128) bytes. Each mobile node can allocate five ($\rho = 5$) data in its storage. Every mobile node issues a top-k query every I (default setting is 30) seconds, where k is set as 25, 50, 75 and 100 based on the two access models (uniform and Zipf-like). In the uniform model, each k is designated with the same probability (25%), and in the Zipf-like model, 25, 50, 75, and 100 are respectively designated with the probabilities of 80%, 10%, 5%, and 5%. respectively.

In this experiments, a node which is randomly selected performs initial collection and distribution at the start of simulation. After 100 seconds, we evaluate the following criteria over 300 queries (i.e., simulation time is $100 + I \times 300$ seconds).

- *Accuracy of query result*: the average ratio of (the number of data acquired by the query-issuer, which are included in the exact result) to (the number of requested data, k).
- *Delay* [second]: the average elapsed time after the query-issuer issues a top-k query until it acquires the result.
- *Query overhead* [Kbytes]: the average volume of query and reply messages (i.e., total volume during the simulation divided by 300). The size of each message is shown in Table 2. In this table, i_1, i_2 and i_3 respectively denote the number of the demanded data, the number of data included in the reply, and the number of data that a node pushes.
- *Initial overhead* [Kbytes]: the total message volume for initial collection and dissemination.

[3] Scalable Networks: makers of QualNet and EXata, the only multi-core enabled network simulation and emulation software. [Online]. Available: http://www.scalable-networks.com/.

Table 2 Message size

Message	Value [bytes]
First query	36
Reiterate query	$36+4 \cdot i_1$
LACMA's query	40
Reply (all methods)	$24+s \cdot i_2$
Push message (LACMA's maintenance)	$32+s \cdot i_3$

6.2 Baselines

We implemented location-based replication methods for comparison with our methods: the expanding ring and bundling methods (graph legends are ER_proposed and B_proposed, respectively). We set δ to 0.0001.

1. LACMA (graph legend is LACMA): Data are replicated in a specific grid (replica ratio is also introduced), and the replicas are maintained to bind a specific grid. When a node leaves its own grid, it pushes the data that should be bound to the grid, and deletes the data from its storage (graph legend "LACMA_M" represents the average maintenance overhead per query interval (30 seconds)). Nodes receiving the data replicate the data if they have available storage. In the query processing, the query-issuer floods a message into its own grid to acquire the top-k result. However, when the query-issuer cannot acquire the result by a pre-determined time that is computed from the grid size, it gives up to acquire the exact result. In this experiments, the grid size is determined based on the access frequency of k and the number of nodes.
2. Reiterate LACMA (graph legend is ReLACMA): After acquiring the result by using LACMA, the query-issuer repeatedly sends a reiterate query in the same way as our methods until the exact result is acquired. The number of hop counts of the first reiterate query is $\lceil \frac{grid\ width \cdot \sqrt{2}}{R} \rceil$, and then it increases by 1.

In addition, we implemented two replication strategies for comparison with our replication strategy.

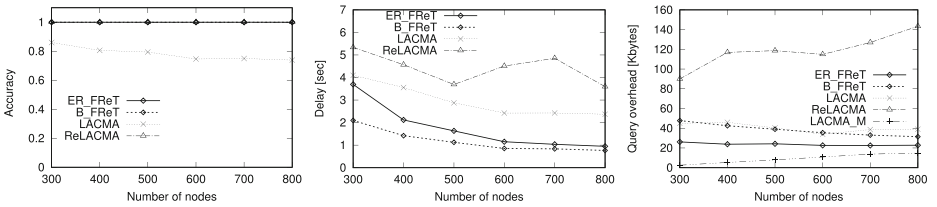
1. Uniform allocation: All the k_K best data are allocated with the same probability.
2. Square root allocation: Data are allocated based on the replication ratio proposed in [3].

In two strategies, we employ the proposed methods for query processing (graph legends for the expanding ring and bundling method with uniform allocation and the expanding ring and bundling methods with square root allocation are ER_UNI, B_UNI, ER_SQRT, and B_SQRT, respectively).

6.3 Experimantal results

6.3.1 Efficiency of our framework

First, we examine the efficiency of our framework. In this section, we vary the number of nodes, the maximum velocity of nodes, and the query interval. Moreover, we check the performance in the case of that the access models are different.



(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 3 Impacts of number of nodes in uniform access model

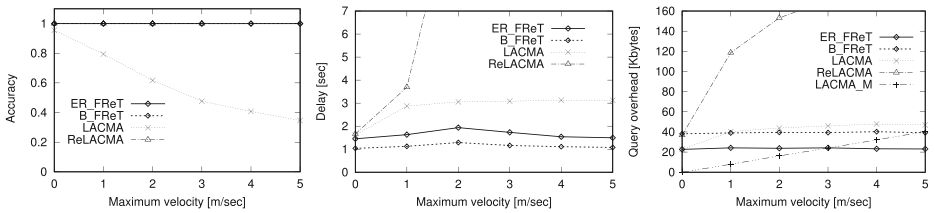
Impact of number of nodes The number of nodes mainly affects the number of neighbor nodes. If the number of nodes is large, the query-issuer can retrieve the result from nearby nodes with a high probability. That is, if the number of nodes increases, the overhead and the delay should decrease. Figure 3 shows the experimental result by varying the number of nodes M in the uniform access model. Figure 3a, b, and c show the accuracy of query result, the delay, and the query overhead, respectively.

From Fig. 3a, we can see that the proposed top-k query processing methods achieve perfect accuracy of query result. On the other hand, LACMA without reiterate query cannot achieve perfect accuracy of query result due to the movement of nodes, the inhomogeneous density of nodes, and packet losses. From Fig. 3b, the bundling method achieves the smallest delay among all methods. This is because the bundling method expands the search area to the area that contains the result with a high probability at once. The expanding ring method also achieves a small delay because the search area is small though the number of transmitted queries is large. When the number of nodes is large, the delay in the expanding ring method is similar to that in the bundling method. This is because as the number of nodes increases, the probability that the result is acquired within 1 hop increases. The delay in LACMA is larger than that in the proposed methods, because, since the query-issuer often cannot acquire the result, it has to wait for the pre-determined time limit. From Fig. 3c, the expanding ring method achieves the smallest query overhead because of acquiring the result from the minimum number of nearby nodes. The bundling method keeps similar overhead to LACMA though it achieves the perfect accuracy of query result. The query overhead of reiterate LACMA is significantly large, because in LACMA, nodes delete their own replicas when they move out from their grids, and thus the search range becomes larger. Moreover, in LACMA, the maintenance cost increases as the number of nodes increases.

From this result, we can confirm that our framework efficiently process top-k query processing regardless of the number of nodes.

Impact of velocity If the velocity increases, the network topology frequently changes in a small time duration. The location-based approach may need a large maintenance cost due to the movement of nodes. Figure 4 shows the experimental result by varying the maximum velocity v in the uniform access model. In the graphs, we show result of ReLACMA where the maximum velocity is equal to or less than two because Reiterate LACMA does not work well when the maximum velocity is larger than two.

From Fig. 4a, we can see that the proposed methods achieve perfect accuracy of query result even if the maximum velocity increases. On the other hand, as the velocity increases, the accuracy of query result decreases in LACMA because many nodes often move out from their grids. From Fig. 4b, the proposed methods achieve better performance than LACMA. Reiterate LACMA takes a long delay to acquire the result because the search range and the



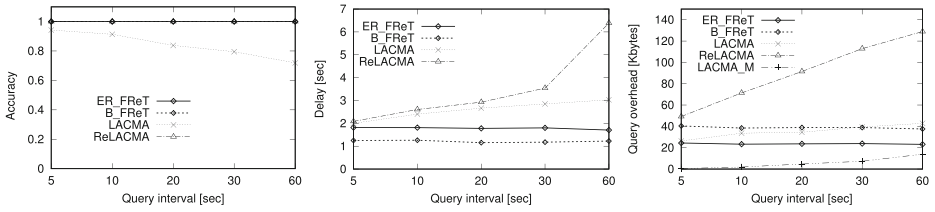
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 4 Impacts of velocity in uniform access model

number of reiterate queries become large. From Fig. 4c, we can see that the query overhead in the proposed methods is insensitive to the movement of nodes. In LACMA, when nodes are stationary ($v = 0$), the query overhead in LACMA and the expanding ring method with our framework are the smallest. Replicas in LACMA are allocated uniformly (i.e., geographically optimal allocation), while the search range is restricted to its own grid. Thus, even if nearby nodes that belong to a different grid have necessary data, these nodes do not send a reply, which result in less volume of replies than other methods. On the other hand, the expanding ring method can acquire the result from the minimum number of nearby nodes. As the result, the query overhead in LACMA and that in the expanding ring method with our scheme become similar though their approach are different. If nodes move, the locations of allocated replicas change, and thus, the query overhead increases. When the velocity is large, the maintenance overhead becomes larger than the overhead for query processing. This fact shows that LACMA is less robust against the movement of nodes in terms of the accuracy of query result and the query overhead.

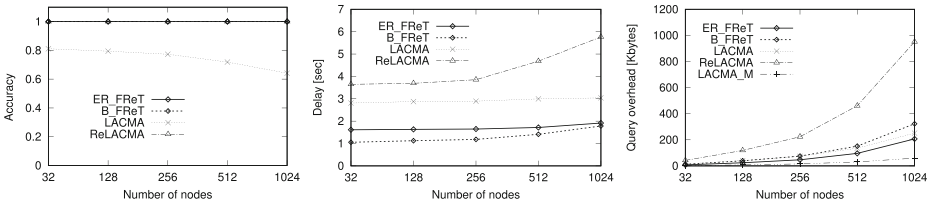
As a result, since our framework requires no network topology information and locations of nodes, the performance of our framework is not affected by the velocity of nodes.

Impact of query interval The query interval affects the ratio between query overhead and maintenance cost. If the query interval is large, the unnecessary maintenance cost happens because mobile nodes do not often issue top-k queries. We check the effect of the query interval to the strategy that needs maintenance of data. Figure 5 shows the experimental result by varying the query interval I in the uniform access model. From Fig. 5, we can see that the proposed methods constantly keep the high performance, while the performance in LACMA decreases. This is because as the query interval increases, the simulation time increases, and thus, nodes move out from their grid more often. As a result, replicas often cannot be bound to a specific grid.



(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 5 Impacts of query interval in uniform access model



(a) Accuracy of query result (b) Delay (c) Query overhead

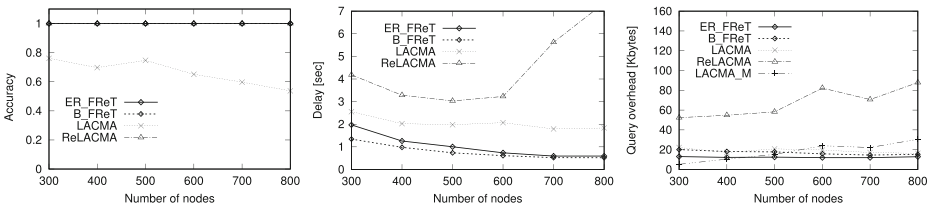
Fig. 6 Impacts of data size in uniform access model

Impact of data size As the size of data increases, the size of reply message increases. Furthermore, when the size of data is large, the delay increases because packet losses frequently happens. Figure 6 shows the experimental result by varying the data size s in the uniform access model. From Fig. 6, we can see that the proposed methods achieve high performance in terms of all metrics. As the size of data increases, LACMA decrease the accuracy of query result. This is because LACMA cannot receives necessary data items due to packet losses. Our methods keep high performance even if the size of data is large.

Impact of access model In the case that access model is different, the efficiency may change. We check the performance of our scheme if the access model changes. Figures 7, 8, 9, and 10 show the experimental result in the Zipf-like access model.

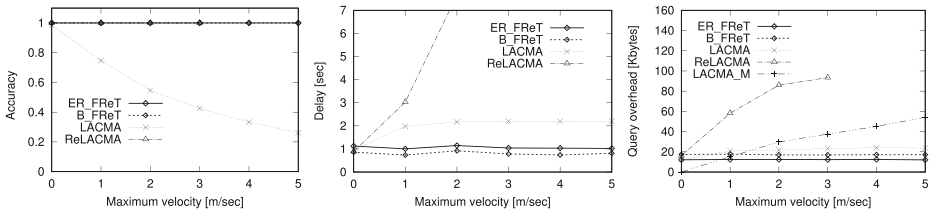
From Fig. 7a, the proposed methods achieve the perfect accuracy of query result even if the access model follows the Zipf-like access model. On the other hand, in LACMA, the accuracy of query result is lower than that in the uniform access model. This is because the grid for data with 25th highest scores becomes very small, and thus, nodes frequently move out from their grids. From Fig. 7b, the delays in the expanding ring and the bundling methods show more similar performance than that in the uniform access model. This is because, since small k is frequently designated in the Zipf-like access model, the query-issuer can likely acquire the result only from very nearby nodes. In ReLACMA, the delay increases when the number of nodes is large. This is because the grid size becomes smaller as the number of nodes increases, and thus, nodes frequently move out from their grids. From Fig. 7c, the query overheads in the Zipf-like access model is smaller than that in the uniform access model because small k is frequently designated. In LACMA, the maintenance overhead is larger than that in the uniform access model for the same reason the accuracy decreases. This means LACMA does not work well in a strongly-biased access model.

Figures 8, 9, and 10 also show that our methods keep the high performance. On the other hand, both LACMA and ReLACMA significantly decrease the performance compared with



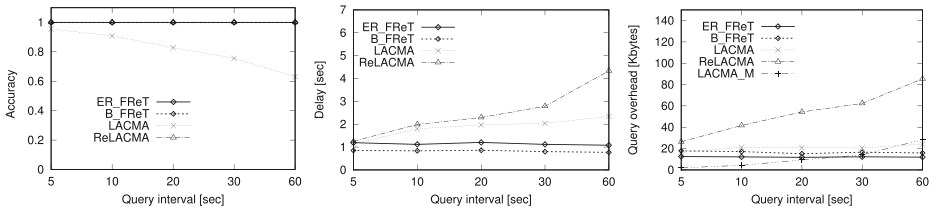
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 7 Impacts of number of nodes in Zipf-like access model



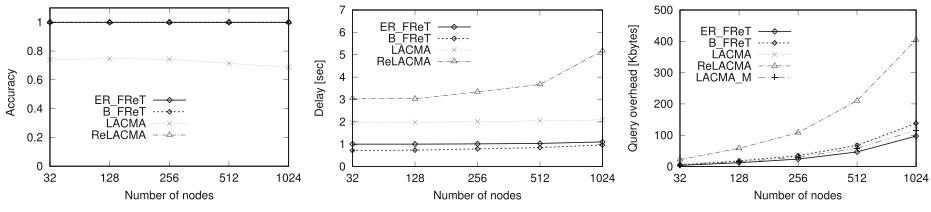
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 8 Impacts of velocity in Zipf-like access model



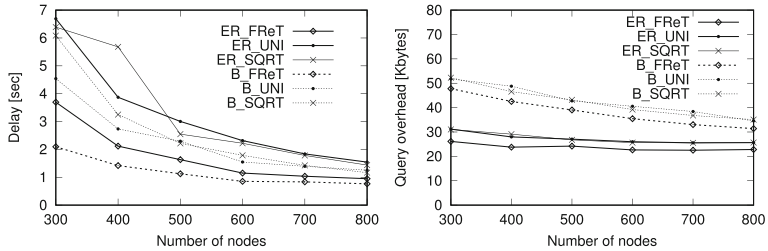
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 9 Impacts of query interval in Zipf-like access model



(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 10 Impacts of data size in Zipf-like access model



(a) Delay (b) Query overhead

Fig. 11 Difference among replica allocation strategies in uniform access model

the result in the uniform access model. Therefore, since our framework can establish the optimal allocation data, it achieves high performance even regardless of the access model.

6.3.2 Efficiency of replication strategy

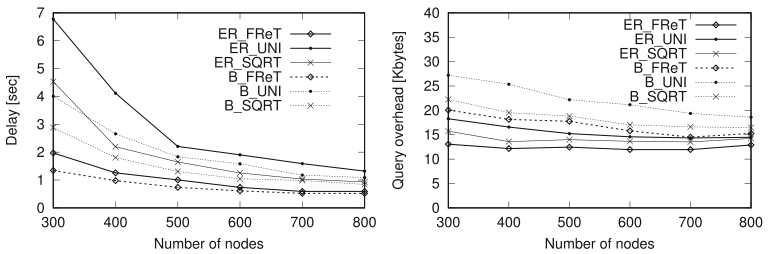
Next, we examine the efficiency of replication strategies. In the above section, we show the performance of our framework that combines our top-k query processing methods and our replication strategy FReT. In this section, we show the result of our top-k query processing methods based on the other replication strategies. Figures 11 and 12 show the result by varying the number of nodes M in the uniform access model and the Zipf-like access model, respectively.

From Fig. 11a, the bundling method with FReT achieves the smallest delay in all methods. The expanding ring method with FReT also achieves the small delay. This shows FReT achieves an appropriate diversity of data. On the other hand, the square root allocation works worse than the uniform allocation. This is because the square root allocation allocates too much data of high-ranked data, and thus the number of reiterate queries increases. From Fig. 11b, the expanding ring method with FReT achieves the smallest query overhead in all methods, and the bundling method with FReT achieves a smaller query overhead than the bundling methods with other replication strategies. The uniform and squire root allocation strategies involve similar query overhead, which shows the average number of hop counts from k data with the highest scores to the query-issuer is similar in both strategies in spite of different replica ratios for data.

From Fig. 12a, FReT achieves the smallest delay. The uniform allocation poorly works in the Zipf-like access model because data with low scores are unnecessarily allocated, while data with high scores should be allocated more. From Fig. 12b, the expanding ring method with FReT achieves the smallest query overhead in all methods, and the bundling method with FReT basically follows it. This is because FReT uses the number of neighbors for data allocation, and thus the query-issuer can acquire the result with a small number of hop counts. On the other hand, the square root allocation unnecessarily allocates data with high scores, and thus the query-issuer receives many duplicate data.

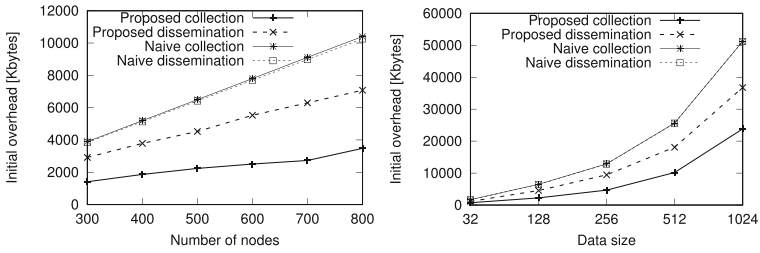
6.3.3 Initial collection and dissemination

We show the performance of the proposed initial collection and dissemination methods compared with a naive method. In the naive collection method, each node employs a simple flooding to send a query message, and then it sends back its own and received data items



(a) Delay (b) Query overhead

Fig. 12 Difference among replica allocation strategies in Zipf-like model



(a) Impacts of the number of nodes (b) Impacts of data size

Fig. 13 Performance of initial collection and distribution

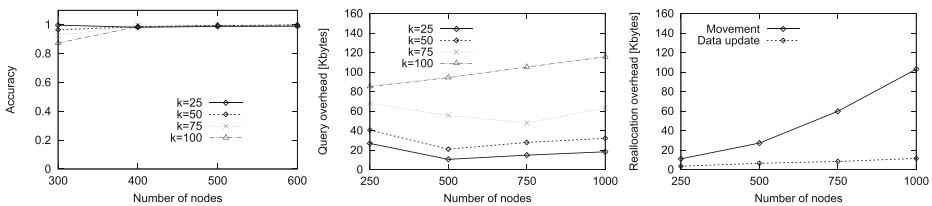
with k_K highest scores. In the naive dissemination method, each node employs a simple flooding to disseminate the collected information.

Figure 13a plots the performance measure in terms of initial overhead by varying the number of nodes M. From this result, our collection and dissemination methods show better performance than the naive methods because of reduction of the number of unnecessary replied data items and message transmission. The overhead in the proposed collection method is smaller than that in the proposed dissemination method, because each node sends less than k_K in the proposed collection method, meanwhile must send k_K data items in the proposed dissemination method. The overheads in the naive collection and dissemination methods are almost same because each node transmits k_K data items. Figure 13b plots the performance measure in terms of initial overhead by varying the data size s. This result also show that our collection and dissemination methods can reduce the initial overhead.

6.3.4 Extension techniques

We evaluate performance for extension techniques, i.e., location-based allocation and data update. In this experiment, we evaluate the accuracy of query result when the query-issuer searches for k data items from its and its neighbor grids. We set that *margin* is 10 and data size is 32 bytes. The query message includes grid IDs to search for data items from the specified grids. From Fig. 14a, the location-based allocation keeps high accuracy without sending reiterate queries.

The reason that it does not achieve the perfect accuracy is packet losses and movements of nodes. From this result, the location-based allocation works well for retrieving the top-k result.



(a) Accuracy of query result (b) Query overhead (c) Reallocation overhead

Fig. 14 Performance of extension techniques

From Fig. 14b, as the number of nodes increases, the overhead decreases except for the case that $k = 100$ because the search range becomes smaller. When the number of nodes further increase, the number of nodes that transmits the message and replied data items increase. Thus, the overhead increases. In the case that $k = 100$, the overhead constantly increases. This is because the search range gets smaller as the number of nodes increases, but the amount of replied data items increases.

From Fig. 14c, the overheads related to movement of nodes and deletion and generation of data items increase as the number of nodes increases. This is because the reallocation frequent occurs when the number of nodes is large. When the number of nodes increases, the size of grids becomes small. The movement of nodes leaving from their grids frequently occurs, and thus the reallocation overhead becomes large. Overhead for data updates is not large compared with query overhead and overhead for movements. Thus, our approach can handle data updates with a small maintenance cost.

7 Conclusion

In this article, we proposed a framework for efficiently processing top-k queries with replication in MANETs. The proposed replication strategy for processing top-k query, FReT, achieves good replica allocation to acquire the top-k result from a small number of nodes. FReT involves no maintenance cost due to nodes' movement. In the proposed top-k query processing method, the query-issuer repeatedly sends a query message until it acquires the perfect top-k result, by increasing the TTL that defines the search area by the number of hop counts. The increase in the TTL is based on two complementary approaches: the expanding ring and bundling methods. The expanding ring method can achieve the smallest number of nodes that need to be accessed for acquiring the top-k result, while the bundling method achieves small delay. Both methods can guarantee the exact accuracy of the query result. We evaluated these methods through experiments which took into account the effect of the physical layer such as radio interference. The experimental results showed that our frameworks achieves better performance than existing methods. The results also show that the expanding ring method achieves small overhead, while the bundling method achieves small delay.

Acknowledgements This research is partially supported by the Grant-in-Aid for Scientific Research (A)(JP16H01722), Scientific Research (S)17H06099, and Young Scientists (B)(JP15K21069).

References

1. Amagata D, Sasaki Y, Hara T, Nishio S (2013) A robust routing method for top-k queries in mobile ad hoc networks. In: MDM, pp 251–256
2. Börzsöny S, Kossmann D, Stocker K (2001) The skyline operator. In: ICDE, pp 421–430
3. Cohen E, Shenker S (2002) Replication strategies in unstructured peer-to-peer networks. In: SIGCOMM, pp 177–190
4. Dimakis AG, Godfrey PB, Wu Y, Wainwright MJ, Ramchandran K (2010) Network coding for distributed storage systems. IEEE TIT 56(9):4539–4551
5. Fiore M, Casetti C, Chiasserini C (2011) Caching strategies based on information density estimation in wireless ad hoc networks. IEEE TVT 60(5):2194–2208
6. Hagihara R, Shinohara M, Hara T, Nishio S (2009) A message processing method for top-k query for traffic reduction in ad hoc networks. In: MDM, pp 11–20
7. Hara T (2001) Effective replica allocation in ad hoc networks for improving data accessibility. In: INFOCOM, pp 1568–1576

8. Hara T, Hagihara R, Nishio S (2010) Data replication for top-k query processing in mobile wireless sensor networks. In: SUTC, pp 115–122
9. Ilyas I, Beskales G, Soliman M (2008) A survey of top-k query processing techniques in relational database systems. ACM CSUR 40(4):11
10. Jiang H, Cheng J, Wang D, Wang C, Tan G (2012) A general framework for efficient continuous multidimensional top-k query processing in sensor networks. IEEE TPDS 23(9):1668–1680
11. Kong J, Lu S, Zhang L, Zerfos P, Luo H (2001) Providing robust and ubiquitous security support for mobile ad hoc networks. In: ICNP, IEEE Computer Society, pp 0251–0251
12. Lee S, Wong S, Lee K (2011) S. Lu. Content management in a mobile ad hoc network Beyond opportunistic strategy. In: INFOCOM, pp 266–270
13. Lu Z, Sun X, La Porta T (2016) Cooperative data offloading in opportunistic mobile networks. In: INFOCOM, pp 1–9
14. Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and replication in unstructured peer-to-peer networks. In: Supercomputing, pp 84–95
15. Ni S, Tseng Y, Chen Y, Sheu J (1999) The broadcast storm problem in a mobile ad hoc network. In: Mobicom, pp 151–162
16. Padhariya N, Mondal A, Goyal V, Shankar R, Madria S (2011) Ecotop: an economic model for dynamic processing of top-k queries in mobile-p2p networks. In: DASFAA, pp 251–265
17. Padmanabhan P, Gruenwald L, Vallur A, Atiquzzaman M (2008) A survey of data replication techniques for mobile ad hoc network databases. VLDB J 17(5):1143–1164
18. Perkins CE, Royer EM (1999) Ad-hoc on-demand distance vector routing. In: Mobile Computing Systems and Applications (WMCSA), IEEE, pp 90–100
19. Ryeng N, Vlachou A, Doulkeridis C, Nørvåg K (2011) Efficient distributed top-k query processing with caching. In: DASFAA, pp 280–295
20. Sasaki Y, Hagihara R, Hara T, Shinohara M, Nishio S (2010) A top-k query method by estimating score distribution in mobile ad hoc networks. In: Advanced Information Networking and Applications Workshops (WAINA), pp 944–949
21. Sasaki Y, Hara T, Ishikawa Y (2018) Top-k query processing with replication strategy in mobile ad hoc networks. In: MDM, pp 217–226
22. Sasaki Y, Hara T, Nishio S (2011) Two-phase top-k query processing in mobile ad hoc networks. In: Network-based Information Systems (NBiS), pp 42–49
23. Sasaki Y, Hara T, Nishio S (2014) Top-k query processing for replicated data in mobile peer to peer networks. System and Software 92:45–58
24. Vlachou A, Doulkeridis C, Nørvåg K (2012) Distributed top-k query processing by exploiting skyline summaries. Distributed and Parallel Databases 30(3-4):239–271
25. Vlachou A, Doulkeridis C, Nørvåg K, Kotidis Y (2012) Peer-to-peer query processing over multidimensional data. Springer Science & Business Media
26. Wu M, Xu J, Tang X, Lee W-C (2007) Top-k monitoring in wireless sensor networks. IEEE TKDE 19(7):962–976
27. Zou L, Chen L (2008) Dominant graph: an efficient indexing structure to answer top-k queries. In: ICDE, pp 536–545

Yuya Sasaki received the B.E., M.E, and Ph.D degrees from Osaka University, Japan, in 2009, 2011, and 2014, respectively. He is currently an Assistant Professor in Graduate school of Information Science and Technology, Osaka University. His research interests include database systems, graph data processing, and data mining.

Takahiro Hara received the B.E, M.E and Dr.E degrees in Information Systems Engineering from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is a full Professor of the Department of Multimedia Engineering, Osaka University. His research interests include database systems, mobile computing systems, Web mining, and mobile ad hoc networks.

Yoshiharu Ishikawa is a professor in Graduate School of Informatics, Nagoya University. He received B.S., M.E., and Dr. Eng. degrees from University of Tsukuba in 1989, 1991, and 1995, respectively. His research interests include spatio-temporal databases, mobile databases, sensor databases, data mining, information retrieval, and e-science.