Top-k Query Processing with Replication Strategy in Mobile Ad Hoc Networks

Yuya Sasaki †, Takahiro Hara †, Yoshiharu Ishikawa ‡

† *Graduate school of Infsormation Science and Technology, Osaka University, Japan*
‡ *Graduate school of Information Science, Nagoya University, Japan*
{sasaki, hara}@ist.osaka-u.ac.jp, ishikawa@i.nagoya-u.ac.jp

***Abstract*—In this paper, we propose a method that fully combines top-k query processing with replication strategy in mobile ad hoc networks (MANETs). The goal is to acquire perfect accuracy of query results with a minimal overhead and delay. Currently, no replication strategy achieves efficient allocation of replicas for top-k queries, and no top-k query processing guarantees perfect accuracy of query results in MANETs. We propose a new replication strategy $FReT$ (topology-Free Replication for Top-k query) and new top-k query processing methods. FReT advantages efficient top-k query processing from limited search area even if mobile nodes move. In our top-k query processing method, the search area gradually increases until receiving an exact answer. We demonstrate, through extensive simulations, that our approaches function well in terms of small delay and overhead.**

***Index Terms*—top-k query, replication, mobile ad hoc networks**

I. Introduction

Wireless communication technologies and computer devices have grown remarkably in the past several decades. These developments bring us a new network concept *mobile ad hoc network (MANET)* [1], [2]. MANETs are opportunistic networks, and composed of autonomous mobile nodes which can communicate each other without any infrastructures. When a mobile node sends to 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 situation such as a disaster site and a military field. MANETs have some important factors to be widely used such as routing protocols [3] and security [4]. One of the important factors is *query processing*, but it has not been studied well. In this paper, 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. These fundamental queries have been developed in many important applications such as (i) Web and Internet search engines, for word-occurrence and text-based relevance [5]; (ii) wireless sensor networks (WSNs), for detection of statistical outliers [6]; and (iii) peer-to-peer (P2P) networks, for sharing interesting contents [7]. Top-k queries are also available in the field of MANETs. For example, in a disaster area where the informational infrastructure (e.g., the Internet) is disabled, numerous mobile rescuers can input on-site information on victims and buildings into their computers, and search for seriously-injured victims and badly-damaged buildings. In this situation, due to a shortage of supplies or manpower (e.g., the number of ambulances is 10 or the number of rescuers is 50), it is desirable to acquire only important information on the most seriously-injured victims or badly-damaged buildings. However, few studies have investigated the potential of top-k query processing in MANETs, though it is a promising application involving a number of challenging issues.

To process top-k queries in MANETs, if 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 basically needs to send a query message to all nodes in the entire network, because the query-issuer does not know which nodes have data items included in an exact answer. However, this procedure is quite inefficient. To avoid this inefficient procedure, we exploit *replication*; each node replicates data items retained by other nodes into its storage. Replication is promising in a MANET environment [8] because it has the capability to improve data availability and limit a search area. By using replication, the query-issuer receives the answer from few nodes. Currently, no study effectively combines top-k query processing with replication in MANETs.

Therefore, we propose a novel top-k query processing method, with replication strategy in MANETs, which is capable of guaranteeing the exact query results in a limited search area. Mobile nodes replicate data items based on our proposed replication strategy called *FReT* (topology-Free Replication for Top-k query). In FReT, the ratio of the number of allocated replicas (*replication ratio*) and combination of replicas retained by each node (*replica combination*) are determined to efficiently acquire the exact answer. Moreover, FReT has no maintenance costs due to the movement of the nodes. 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.

The contributions of this paper are as follows:

- Prior work does not effectively acquire an exact answer in environments where mobile nodes hold replicas. The proposed approach is the first method which combines top-k query processing and 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 changes.
- We demonstrate, through extensive simulations taking into account the effect of the physical layer, that our approaches function well in terms of small delay and overhead.

The remainder of this paper is organized as follows. Section II introduces the related work. Section III provides preliminaries. Section IV presents the proposed method. Section V summarizes the results obtained in the simulation experiments. Section VI concludes the paper.

II. Related work

There are 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.

A. 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. In [10], 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 [11], [12], the authors proposed a caching method for top-k query processing in MANETs[1]. This method caches undue data items with low scores, and it is difficult to determine an appropriate random number without a knowledge of scores of all data items. Thus, this caching method is far from optimal Moreover, the authors [12] 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 [13], 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 [14], the authors proposed a form of location-based replication called 'location-aided content management architecture' (LACMA), which supports searches 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 [11], [12] 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 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 cost.

B. 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 [15], the authors proposed a method for filtering out unnecessary data items by using *skylines*. However, these strategies consider neither packet collision nor movement of nodes, and thus cannot be directly adopted to MANETs. 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 [16], the authors proposed a method which constructs a top-k filter by utilizing a dominant graph which was discussed in [17] as a data structure for efficient top-k query processing in centralized databases.

A few works such as [18]–[20] are extant studies that address the issue of top-k query processing in MANETs. The method proposed in [18] assumed an economic scheme (e.g., virtual currency) with a fundamentally different assumption from that of our study. In the methods proposed in [19], [20], 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. 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.

III. Preliminaries

Table I summarizes the symbols used in the paper.

[1] Although the proposed method in [11] is named a replication method, this method is essentially a caching protocol because each node stores data items when it receives data items.

A. System model

The system environment is assumed to be a MANET in which mobile nodes retrieve k data items with the highest scores (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, and the scores of data items can be calculated from a query condition. k is selected from K different values k_i $(i = 1, \cdots, K)$ with a probability of kq_i. For simplicity, all nodes designate k_i with the same probability (i.e., use a given same access model). 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 paper, we focus on only one pair of query condition and scoring function, 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 score 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 hold the same type of device. Moreover, while we assume for simplicity that mobile nodes issue only one specific type of query, mobile nodes use only a small portion of their storage for replication (i.e., ρ is very small).

B. Problem formulations

Since problems of top-k query processing, replication, and MANETs are intricately intertwined, we analyze the problems in this subsection.

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.

TABLE I: Symbols

Symbol	Meaning
k	Number of requested data items
k_i	K different values of k $(i = 1, ..., K)$
kq_i	Probability that k_i is designated
k_{max}	The maximum k, i.e., $\max_{1 \leq i \leq K} k_i$
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
r_i	Replication ratio for d_i

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 approach, 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). (1) is the most important measure because if (1) is very low, small (2) and (3) have no meaning, so a better goal is to keep perfect (1) and optimize (2) and (3) as much as possible. Moreover, (2) and (3) have a trade-off and should be considered together. To achieve both small (2) and (3), we try to reduce the number of nodes that must be accessed in order to acquire the top-k result. Because, if the number of nodes that must be accessed is small, the query-issuer can acquire the top-k result from only nearby mobile nodes. Therefore, in this paper, we try to determine the optimal replica allocation to reduce $An(\mathbf{kq})$, the average number of nodes that must be accessed in order to acquire the top-k result over the entire system. $An(\mathbf{kq})$ is calculated based on the following equation:

$$An(\mathbf{kq}) = \sum_{i=1}^{K} kq_i \cdot An(k_i) \tag{1}$$

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

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 [9] 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 replication ratio for d_i based on the following equation:

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

where q_i is the access rate for d_i, k_{max} is the maximum k, and l ($\geq \frac{1}{M \cdot \rho}$) and u ($\leq \frac{1}{M}$) 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 scores are more frequently accessed than those with lower scores. Therefore, the replication ratio 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 replication ratio 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 scores, 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) 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:

$$cost = \Sigma_{x \in m} \Sigma_{i=1}^{K} k q_i \Sigma_{j=1}^{k_i} dist(x, d_j) \tag{4}$$

where $dist(x, d_j)$ denotes the distance (basically, the Euclidean distance) 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.

IV. Proposed methods

In this section, we describe the proposed replica allocation strategy, the initial collection and distribution methods, and the top-k query processing method.

A. FReT: Replica allocation strategy

Replica combination. In top-k queries, data items with similar ranks are accessed together with high probability for a given k. Therefore, each node allocates ρ replicas with consecutive ranks (i.e., d_1 through d_ρ, $d_{\rho+1}$ through $d_{2 \cdot \rho}$, $\cdots$, and $d_{\lceil \frac{k_{max}}{\rho} \rceil \cdot \rho - \rho + 1}$ through $d_{k_{max}}$). We call a set of ρ replicas with consecutive ranks a *replica combination*. The number of replica combinations is $\lceil \frac{k_{max}}{\rho} \rceil$, and each data item is replicated based on the *replica combination ratio* that is defined as rc_i (e.g., rc_1 is the replication ratio for d_1 to d_ρ). In FReT, the replica combination ratio is calculated in order to achieve the efficient allocation of replicas.

Replication ratio. The query-issuer can acquire the top-k result from a small number of nodes if the efficient replica combination ratio is established. The number of nodes that are accessed depends on the number of neighbors for each node and a hop count from a given node. When the hop count and/or the number of neighbors are large, the number of nodes that are accessed increases. Given a hop count hop, we calculate the expected number of nodes that are accessed n_h based on the following equation:

$$n_h = 1 + n_{avg} \times \sum_{j=0}^{hop} j \tag{5}$$

where n_{avg} denotes the average number of neighbors for all nodes (n_{avg} is estimated after an initial collection described later). In this equation, we assume that nodes uniformly exist, and all nodes have the same number of neighbors (n_{avg}). Thus, n_h uniformly increases as hop increases. 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 $P(k_i, n_h)$ as the probability that the top-k result is acquired by searching n_h nodes:

$$\begin{aligned} P(k_i, n_h) \quad = \quad & 1 - (\overline{rc_1} + \cdots + \overline{rc_{\lceil \frac{k_i}{\rho} \rceil}}) \\ & + (\overline{rc_1 + rc_2} + \cdots + \overline{rc_{\lceil \frac{k_i}{\rho} \rceil - 1} + rc_{\lceil \frac{k_i}{\rho} \rceil}}) \\ & + \cdots \\ & + (-1)^{\lceil \frac{k_i}{\rho} \rceil} \cdot \overline{rc_1 + \cdots + rc_{\lceil \frac{k_i}{\rho} \rceil}} \end{aligned} \tag{6}$$

where $\overline{x}$ denotes $(1 - x)^{n_h}$. This equation calculates the probability of a complementary event that a given node cannot access replica combinations: rc_1 to $rc_{\lceil \frac{k_i}{\rho} \rceil}$ based on Inclusion-exclusion principle. If n_h is less than $\lceil \frac{k_i}{\rho} \rceil$, $P(k_i, n_h)$ becomes 0. This is because we need to access at least $\lceil \frac{k_i}{\rho} \rceil$ nodes to obtain k_i data items. $P(k_i, n_h)$ increases as n_h (i.e., hop) increases because the probability decreases that the node cannot access necessary replicas. Of course, a smaller n_h that achieves a larger $P(k_i, n_h)$ is better. Thus, the optimal hop is determined to achieve the smallest $\frac{n_h}{P(k_i, n_h)}$. In addition, $P(k_i, n_h)$ increases when data items d_1 through d_{k_i} are

Algorithm 1 FReT algorithm

Input: ρ, ann, kq and δ
Output: rc
1: $rc_{num} = \lceil \frac{k_{max}}{\rho} \rceil$
2: $rc_i = \frac{1.0}{rc_{num}} (i = 1, \cdots, rc_{num})$
3: **while not** all patterns of rc are done **do**
4: **for** $i = 1, \cdots, K$ **do**
5: **for** $hop = 1, \cdots$ **do**
6: $n_{h_i} = ann \cdot \sum_{j=1}^{hop} j$
7: $An(k_i)_{hop} = \frac{n_{h_i}}{P(k_i, n_{h_i})}$
8: **if** $hop \neq 1$ **and** $An(k_i)_{hop} \geq An(k_i)_{hop-1}$ **then**
9: $An(k_i) = An(k_i)_{hop-1}$ and **break**
10: **end if**
11: **end for**
12: **end for**
13: $An(\mathbf{kq}) = \sum_{i=1}^{K} kq_i \cdot An(k_i)$
14: **if** $minAn > An(\mathbf{kq})$ **then**
15: $minAn = An(\mathbf{kq})$, and $result \leftarrow rc$
16: **end if**
17: Calculate next rc based on δ
18: **end while**

frequently replicated (i.e., rc_1 to $rc_{\lceil \frac{k_i}{\rho} \rceil}$). However, if we prioritize only small k, the diversity of replicas decreases because rc with small subscript significantly increases. Therefore, we should calculate n_h for every k_i, n_{h_i}, to totally achieve a small number of nodes that must be accessed in the entire MANET. For this aim, we calculate $An(\mathbf{kq})$ based on the following equation:

$$An(\mathbf{kq}) = \sum_{i=1}^{K} (kq_i \cdot \frac{n_{h_i}}{P(k_i, n_{h_i})}). \quad (7)$$

If $An(\mathbf{kq})$ is minimized, the query-issuer can acquire necessary data items by searching a small number of nodes with high probability. Therefore, in FReT, the replica combination ratio is established so as to achieve minimum $An(\mathbf{kq})$.

Algorithm 1 shows pseudo-code to establish $An(\mathbf{kq})$. In this algorithm, δ denotes a parameter value which shreds rc evenly.[2] A smaller hop value which achieves minimum $An(k_i)$ for each i is calculated for a given rc, and then $An(\mathbf{kq})$ is calculated by summing all $An(k_i)$ (lines 4 to 13). We try to calculate all patterns of rc (recall that $rc_i \geq rc_{i+1}$) (line 17). To reduce the computation cost, the rc_i with a same access ratio are calculated at once. Finally, after all patterns of rc are checked, rc with minimum $An(\mathbf{kq})$ is returned.

FReT is a high robust replication strategy because the replication ratio does not sensitively change unless the network topology significantly changes. Our algorithm establishes the replication 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.

[2]A smaller δ can calculate more precise replica combination ratio, but increases the computation cost. We can control δ based on a computational power of mobile nodes.

B. Initial collection and distribution

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, the node which is the first query-issuer, m_c, becomes a *coordinator*, and transmits an *initial message* to acquire the top-k result and the node positions. To reduce the overhead for initial collection as possible, we employ an existing protocol such as a top-k query processing method proposed in [21] and a location-based flooding method proposed in [22].[3] After m_c completes this process, it knows the top-k result and calculates the average number of their 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. Then, m_c disseminates the top-k result and the average number of their neighboring nodes by an existing flooding technique. Receivers store the data identifiers and scores of k_{max} data items with the highest scores (or top-k result) and the average number of their neighboring nodes. Then, each node calculates the replica combination ratio based on FReT, and allocates replicas. Here, receivers are aware of k_{max} data items with the highest scores, but since their storage has a limit, they store only the data identifiers and scores of data items instead of whole data items, except for their allocated replicas.

C. Top-k query processing

In this subsection, we describe two top-k query processing methods; the *expanding ring* and *bundling* methods. In both methods, the query-issuer initially sends a *first query*, and then repeatedly sends a *reiterate query* until the top-k result is acquired. The difference between the two methods lies in the means of setting the TTL.

The processing of the first query and that of the reiterate query are shown in Algorithms 2 and 3. The first query message, FQ, includes the identifier of the query-issuer, the identifier of the query, a query condition, k, the list of replicas retained by the query-issuer ($list_{qi}$). 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 items, the identifier of the sender node, the position of the sender node, and the TTL. Both reply messages include the identifier of the query-issuer, the identifier of the query, the identifier of the sender node, the identifier of the sender's parent, and the list of reply data items ($\mathbf{rd}$). In both algorithms, m_q and m_r denote the query-issuer and a node that receives a query message, respectively. Each node overhears these messages in order to reduce the number of reply data items as much as possible (lines 16–18 in Algorithm 2, and lines 31–33 in Algorithm 3).

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 items. This

[3]We do not restrict exiting methods which the coordinator use, but methods without a special assumption and use of previous knowledges are preferable.

Algorithm 2 First query processing

Input: k and a query condition
Output: k data items with the highest scores
if M_q has k data items with the highest scores as its replicas **then**
  Query is over
else
  Query issuer m_q broadcasts FQ
end if
if Node m_r receives FQ **then**
  Stores the information on FQ
  Sets a reply timer as a random value
end if
if m_r expires its reply timer **then**
  for Replicas, d_i held by m_r that are not included $list_{qi}$ and are not overheard **do**
    $\mathbf{rd} \leftarrow \mathbf{rd} \cup d_i$
  end for
  Sends a reply message to the query-issuer
end if
if m_r overhears a reply message **then**
  Stores $\mathbf{rd}$.
end if
if m_q acquires the top-k result **then**
  Query is over
else if m_q waits a maximum reply timer **then**
  Stores the number of nodes replied for the bundling method
  Go to Algorithm 3
end if

Algorithm 3 Reiterate query processing

Input: Identifiers of demanded data items
Output: 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**
      $\mathbf{rd} \leftarrow$ the demanded data items retained by m_r
      Sends a reply message to its parent
      Demanded data items $\leftarrow$ demanded data items - $\mathbf{rd}$
    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 rd_i are not sent to parent node, and are not overheard **do**
    $\mathbf{rd} \leftarrow \mathbf{rd} \cup rd_i$
  end for
  if $\mathbf{rd} \neq \phi$ **then**
    Sends a reply message to its parent
  end if
else if m_r overhears a reply message **then**
  Stores $\mathbf{rd}$
end if
if m_q acquires the top-k result **then**
  Query is over
else if m_q expires a timer determined based on TTL **then**
  Updates RQ
  m_q broadcasts RQ again
end if

is because the combined size of k and the query condition is less than that of all the identifiers of demanded data items, but the query-issuer cannot designate only demanded data items in the reiterate query by designating k and the query condition. In the first query, duplicate reply data items may be sent in reply messages from nodes though the query-issuer has those replicas. Thus, we use $list_{qi}$ to reduce the number of such duplicate reply data items. Second, the latter uses location-based flooding, but the former does not. This is because the reiterate query in which the TTL is large has more chance of employing location-based flooding, but the first query in which the TTL is 1 does not. Thus, in the case of the first query, the location information unnecessarily increases the message size. Finally, the latter sets the TTL to expand the search range, but the former does not set (i.e., the TTL is always 1). It has two reasons; to acquire the top-k result from nearby nodes, and to know the current number of neighboring nodes as soon as possible.

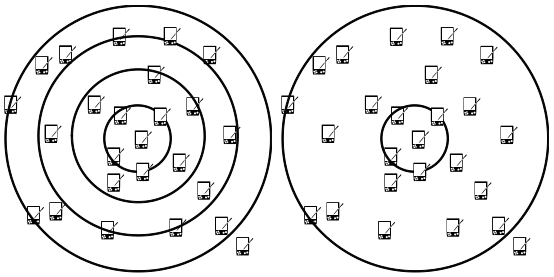
Expanding ring method. In the expanding ring method in unstructured P2P networks proposed in [23], the TTL gradually increases in order to minimize the number of nodes that must be accessed. In our version of method, the TTL of every first query is set as 1, and the TTL of the reiterate query is set as the previous TTL plus 1. The simple modification makes it possible to minimize the number of reply data items. However, when the number of neighbors is small, the number of reiterate queries increases, and thus, the delay may increase.

Fig. 1(a) shows an image of increasing TTL in the expanding ring method. The circles in this figure, which denote the search area, gradually expand.

Bundling method. In the bundling method, the query-issuer sets the TTL based on the current number of neighboring nodes cnn. However, since the query-issuer does not know its cnn in advance, the TTL of the first query is set as 1. After the first query, since the query-issuer now knows its cnn, 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:

$$cnn \times \sum_{j=1}^{TTL} j > \frac{n_{h_i}}{P(k_i, n_{h_i})} \tag{8}$$

where n_{h_i} and $P(k_i, n_{h_i})$ are respectively 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 n_{h_i} nodes in Eq. (7). 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. However, due to an excessive increase of TTL, the



(a) Expanding ring method (b) Bundling method

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

number of reply data items is basically greater than in the expanding ring method.

Fig. 1(b) shows an image of increasing TTL in the bundling method. The TTL of the first reiterate query becomes large at once.

V. Performance Evaluation

In this section, we summarize the results of simulation experiments evaluating performance of our method. 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).[4]

A. Simulation model

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 in the entire network is M (300-800, default setting is 500). The nodes move according to the random-walk model, with a random velocity range of 0.5 to v (0-5, 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 items. The size of a data item is 128 bytes. The score of each data item is randomly determined within a range of 0 to 100. Each mobile node can replicate 5 ($\rho = 5$) data items in its storage.

Every mobile node issues a top-k query every 30 seconds, where k ($k_{max} = 100$ and $K = 4$) 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%.

In this simulation, 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 1000 ($100+30\times300$) seconds).

- *Accuracy of query result*: the average ratio of (the number of data items acquired by the query-issuer, which are included in the top-k result) to (the number of requested data items, 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 II. In this table, i_1, i_2 and i_3 respectively denote the number of the demanded data items, the number of data items included in the reply, and the number of data items that a node pushes.

TABLE II: Message size

Message	Value [bytes]
First query	36
Reiterate query	$36+4 \cdot i_1$
LACMA's query	40
Reply (all methods)	$24+128 \cdot i_2$
Push message (LACMA's maintenance)	$32+128 \cdot i_3$

B. Baselines

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

1) LACMA (graph legend is LACMA): Data items are replicated in a specific grid (replica combination is also introduced). In this simulation, the grid size is determined based on the access frequency of k and the number of nodes. In the query processing, the query-issuer floods a message into its own grid to acquire k data items with the highest scores. However, when the query-issuer cannot acquire the top-k result by a pre-determined time limit that is determined based on the grid size, it gives up to acquire the top-k result. In LACMA, the replicas are maintained to bind a specific grid. When a node leaves its own grid, it pushes the data items which should be bound to the grid, and deletes the data items from its storage (graph legend "LACMA_M" represents the average maintenance overhead per query interval (30 seconds)). Nodes receiving the data items replicate the data items if they have available storage.
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 top-k result is acquired. The TTL 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 FReT.

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

1) Uniform replication: All data items with k_{max} highest scores are replicated with the same probability.
2) Square root replication: Data items are replicated based on the replication ratio proposed in [9].

In all replication strategies including FReT, we employ the proposed methods for query processing (graph legends for uniform replication and square root replication are ER_uni, B_uni, ER_SQRT, and B_SQRT, respectively).

C. Simulation results

1) Top-k query processing: First, we examine the performance of the proposed top-k query processing methods.

Impact of number of nodes. Fig. 2 shows the simulation result by varying the number of nodes M in the uniform access model. From Fig. 2(a), 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. 2(b), the bundling method which we proposed achieves smallest delay among all methods. This is because the bundling method expands the search area to the area that contains the top-k result with high probability at once. The expanding ring method also achieves 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 top-k 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 top-k result, it has to wait for the predetermined time limit.

From Fig. 2(c), the expanding ring method achieves the smallest query overhead because of acquiring the top-k 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.

Impact of velocity. Fig. 3 shows the simulation 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 2 because Reiterate LACMA does not work well when the maximum velocity is larger than 2. From Fig. 3(a), 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 grid.

From Fig. 3(b), the proposed methods achieve better performance than LACMA. Reiterate LACMA experiences significant long delay to acquire the top-k result because the search range and the number of reiterate queries become significantly large.

From Fig. 3(c), 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 FReT 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 items, 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 top-k result from the minimum number of nearby nodes. As the result, the query overhead in LACMA and that in the expanding ring method with FReT 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.

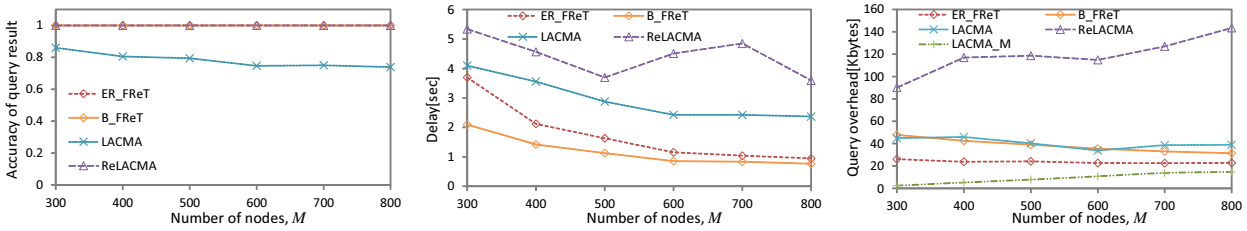
Impact of access model (Zipf-like). Fig. 4 shows the simulation result by varying the number of nodes M in the Zipf-like access model. From Fig. 4(a), the proposed methods achieve the perfect accuracy of query result even if the access model follows the Zipf-like access model. On the other hand, in LACMA, the accuracy of query result is lower than that in the uniform access model. This is because the grid for data items with 25th highest scores becomes very small, and thus, nodes frequently move out from their grids.

From Fig. 4(b), 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 top-k 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. 4(c), 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.

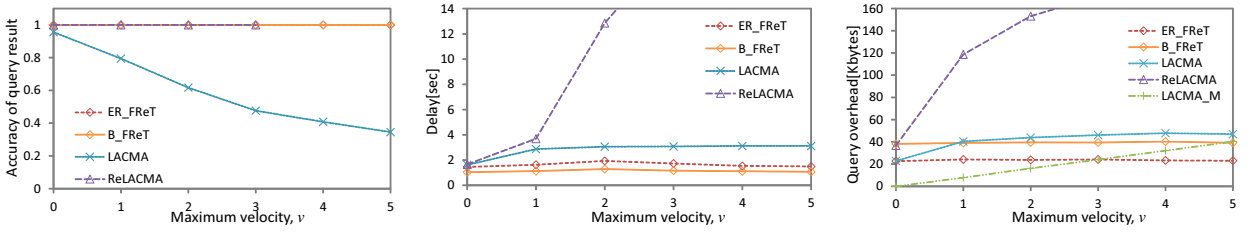
2) Replica allocation strategy: Next, we examine the performance of replica allocation strategies. Figs. 5 and 6 show the simulation result by varying the number of nodes M in the uniform access model and the Zipf-like access model, respectively.

From Fig. 5(a), the bundling method with FReT achieves the smallest delay in all methods. The expanding ring method



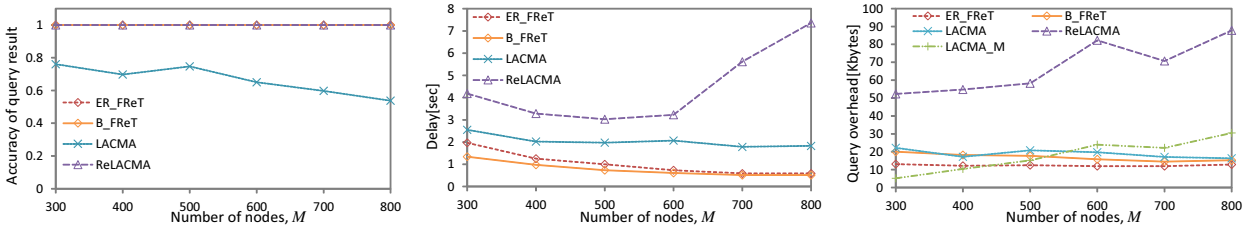
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 2: Impacts of number of nodes in uniform access model



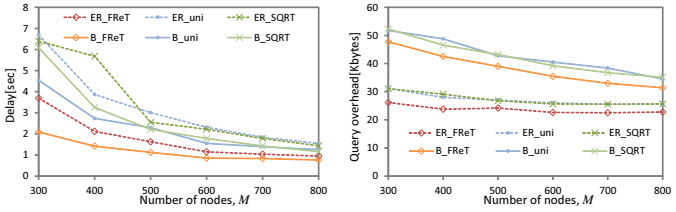
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 3: Impacts of velocity in uniform access model



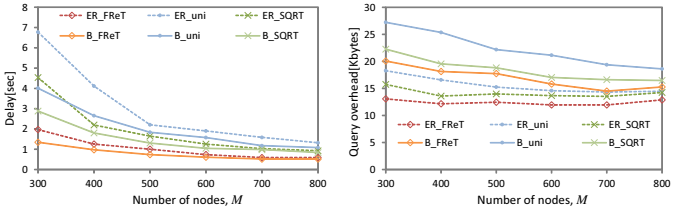
(a) Accuracy of query result (b) Delay (c) Query overhead

Fig. 4: Impacts of number of nodes in Zipf-like access model



(a) Delay (b) Query overhead

Fig. 5: Difference among replica allocation strategies in uniform access model



(a) Delay (b) Query overhead

Fig. 6: Difference among replica allocation strategies in Zipf-like model

with FReT also achieves the small delay. This shows FReT achieves an appropriate diversity of replicas. On the other hand, the square root replication works worse than the uniform replication. This is because the square root replication allocates too much replicas of high-ranked data items, and thus, the number of reiterate queries increases. From Fig. 5(b), the expanding ring method with FReT achieves the smallest query overhead in all methods, and the bundling method with FReT achieves smaller query overhead than the bundling methods with other replication strategies. The uniform and squire root replication strategies involve similar query overhead, which shows the average hop count from k data items with the highest scores to the query-issuer is almost similar in both strategies in spite of different replication ratios for data items.

From Fig. 6(a), FReT achieves the smallest delay. The uniform replication poorly works in the Zipf-like access model because data items with low scores are unnecessarily replicated, while data items with high scores should be replicated

more. From Fig. 6(b), 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 takes into account the number of neighbors for data replication, and thus, the query-issuer can acquire the top-k result with small hop count. On the other hand, the square root replication unnecessarily replicates data items with high scores, and thus the query-issuer receives many duplicate data items.

3) Summary: From the above results, we can see that the proposed methods achieve perfect accuracy of query result with small overhead and delay in all situations. In addition, FReT achieves higher performance than the location-based replication strategy in MANETs and other replication strategies.

VI. Conclusion

In this paper, we proposed a top-k query processing method with replication in MANETs. The proposed replication strategy for 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 hop count. 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 simulations which took into account the effect of the physical layer such as radio interference. The simulation results show that FReT achieves better performance than existing replication strategies. The results also show that the expanding ring method achieves small overhead, while the bundling method achieves small delay.

In this paper, we simplified some assumptions, e.g., no data update, specific query condition, and all nodes use same access model and hold same device. These assumptions are not always true in a real environment. Thus, we plan to extend our methods to work without these assumptions.

Acknowledgement

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] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE TIT*, vol. 56, no. 9, pp. 4539–4551, 2010.

[2] Z. Lu, X. Sun, and T. La Porta, "Cooperative data offloading in opportunistic mobile networks," in *INFOCOM*, 2016.

[3] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications (WMCSA)*. IEEE, 1999, pp. 90–100.

[4] J. Kong, S. Lu, L. Zhang, P. Zerfos, and H. Luo, "Providing robust and ubiquitous security support for mobile ad hoc networks," in *ICNP*. IEEE Computer Society, 2001, pp. 0251–0251.

[5] I. Ilyas, G. Beskales, and M. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM CSUR*, vol. 40, no. 4, p. 11, 2008.

[6] M. Wu, J. Xu, X. Tang, and W.-C. Lee, "Top-k monitoring in wireless sensor networks," *IEEE TKDE*, vol. 19, no. 7, pp. 962–976, 2007.

[7] A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis, "Top-k queries," *Peer-to-Peer Query Processing over Multidimensional Data*, pp. 63–72, 2012.

[8] P. Padmanabhan, L. Gruenwald, A. Vallur, and M. Atiquzzaman, "A survey of data replication techniques for mobile ad hoc network databases," *VLDB Journal*, vol. 17, no. 5, pp. 1143–1164, 2008.

[9] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *SIGCOMM*, 2002, pp. 177–190.

[10] M. Fiore, C. Casetti, and C. Chiasserini, "Caching strategies based on information density estimation in wireless ad hoc networks," *IEEE TVT*, vol. 60, no. 5, pp. 2194–2208, 2011.

[11] T. Hara, R. Hagihara, and S. Nishio, "Data replication for top-k query processing in mobile wireless sensor networks," in *SUTC*, 2010, pp. 115–122.

[12] Y. Sasaki, T. Hara, and S. Nishio, "Top-k query processing for replicated data in mobile peer to peer networks," *Journal of Systems and Software*, vol. 92, pp. 45–58, 2014.

[13] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," in *INFOCOM*, 2001, pp. 1568–1576.

[14] S. Lee, S. Wong, K. Lee, and S. Lu, "Content management in a mobile ad hoc network: Beyond opportunistic strategy," in *INFOCOM*, 2011, pp. 266–270.

[15] A. Vlachou, C. Doulkeridis, and K. Nørvåg, "Distributed top-k query processing by exploiting skyline summaries," *Distributed and Parallel Databases*, vol. 30, no. 3-4, pp. 239–271, 2012.

[16] H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan, "A general framework for efficient continuous multidimensional top-k query processing in sensor networks," *IEEE TPDS*, vol. 23, no. 9, pp. 1668–1680, 2012.

[17] L. Zou and L. Chen, "Dominant graph: An efficient indexing structure to answer top-k queries," in *ICDE*, 2008, pp. 536–545.

[18] N. Padhariya, A. Mondal, V. Goyal, R. Shankar, and S. Madria, "Ecotop: an economic model for dynamic processing of top-k queries in mobile-p2p networks," in *DASFAA*, 2011, pp. 251–265.

[19] Y. Sasaki, R. Hagihara, T. Hara, M. Shinohara, and S. Nishio, "A top-k query method by estimating score distribution in mobile ad hoc networks," in *Advanced Information Networking and Applications Workshops (WAINA)*, 2010, pp. 944–949.

[20] Y. Sasaki, T. Hara, and S. Nishio, "Two-phase top-k query processing in mobile ad hoc networks," in *Network-Based Information Systems (NBiS)*, 2011, pp. 42–49.

[21] R. Hagihara, M. Shinohara, T. Hara, and S. Nishio, "A message processing method for top-k query for traffic reduction in ad hoc networks," in *MDM*, 2009, pp. 11–20.

[22] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom*, 1999, pp. 151–162.

[23] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Supercomputing*, 2002, pp. 84–95.