
Volume 92, June 2014
ISSN 0164-1212
ELSEVIER
The Journal of
Systems and Software

The Journal of Systems and Software 92 (2014) 45–58

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Controversy Corner

Top-k query processing for replicated data in mobile peer to peer networks☆

Yuya Sasaki*, Takahiro Hara, Shojiro Nishio

Osaka University, Graduate School of Information Science and Technology, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

ARTICLE INFO

Article history:
Received 14 September 2012
Received in revised form
20 September 2013
Accepted 20 October 2013
Available online 29 October 2013

Keywords:
Top-k query
Mobile peer to peer network
Data replication

ABSTRACT

In mobile ad hoc peer to peer (M-P2P) networks, since nodes are highly resource constrained, it is effective to retrieve data items using a top-k query, in which data items are ordered by the score of a particular attribute and the query-issuing node acquires data items with the *k* highest scores. However, when network partitioning occurs, the query-issuing node cannot connect to some nodes having data items included in the top-k query result, and thus, the accuracy of the query result decreases. To solve this problem, data replication is a promising approach. However, if each node sends back its own data items (replicas) responding to a query without considering replicas held by others, same data items are sent back to the query-issuing node more than once through long paths, which results in increase of traffic. In this paper, we propose a top-k query processing method considering data replication in M-P2P networks. This method suppresses duplicate transmissions of same data items through long paths. Moreover, an intermediate node stops transmitting a query message on-demand.

1. Introduction

A mobile ad hoc peer-to-peer (M-P2P) network is generally constructed by mobile nodes that have sensing and wireless communication capabilities. Mobile nodes include mobile terminals held by members of rescue operations, excavation activities, or military affair who are engaged in a collaborative work and share the information among them. They also include in-vehicle systems such as car navigation systems that share location dependent information, e.g., traffic, accidents, gas stations, and restaurants around the current position, and information for safe driving. In the former case, because users have to carry mobile terminals, the resources of nodes are basically constrained in terms of computation power, communication bandwidth, battery, and storage. In the latter case, nodes basically have rich resources. In this paper, we assume the former case.

In an M-P2P network with limited resources (in what follows, we simply call it M-P2P network), it is very important that the system middleware supports data management for mobile users so that they can efficiently share information among them. In particular, since the communication bandwidth and battery of nodes are limited, it is important to reduce traffic by acquiring only necessary data when accessing data items. A possible and promising solution is that each node retrieves data using a top-k query, in which data items are ordered by the score of a particular attribute and the node that retrieves data items (*the query-issuing node*) acquires ones with the *k* highest scores (*top-k result*). A rescue effort at a disaster site is a good example of our target applications. In a rescue effort at a disaster site, top-k query is suitable to retrieve necessary data items in short time (small delay). As supplies (e.g., the number of ambulances or amount of medicines) are limited in a rescue effort, rescuers generally want to acquire only specific information such as that on seriously injured victims or the closest victims from rescuers as soon as possible.

In an M-P2P network, network partitioning occurs because the network topology dynamically changes due to movement of mobile nodes. If the network is partitioned, the query-issuing node cannot acquire some data items that are held by mobile nodes in a different partition and included in the top-k result. This means that the accuracy of the query result decreases. To improve data availability at the point of network partitioning in M-P2P networks, the most

☆ *Controversy corner*. It is the intention of the Journal of Systems and Software to publish, from time to time, articles cut from a different cloth. This is one such article. The goal of CONTROVERSY CORNER is both to present information and to stimulate thought and discussion. Topics chosen for this coverage are not just traditional formal discussions of research work; they also contain ideas at the fringes of the field's "conventional wisdom".
These articles will succeed only to the extent that they stimulate not just thought, but action. If you have a strong reaction to the article that follows, either positive or negative, send it along to your editor, at card@software.org.
We will publish the best of the responses as CONTROVERSY REVISITED.

* Corresponding author. Tel.: +81 668794513; fax: +81 668794514.
E-mail addresses: sasaki.yuya@ist.osaka-u.ac.jp (Y. Sasaki), hara@ist.osaka-u.ac.jp (T. Hara), nishio@ist.osaka-u.ac.jp (S. Nishio).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.10.043

promising solution is *data replication* (Cao et al., 2004; Hara, 2001, 2010; Karumanchi et al., 1999; Padmanabhan et al., 2008).

Fig. 1 shows an example where mobile devices held by rescuers of a rescue effort at a disaster site form an M-P2P network. Each rescuer collects information on the environment and victims by using several kinds of sensing devices and shares the information among rescuers. When the rescuer in the upper-left corner wants to find the three most seriously injured victims in the entire area (e.g., the number of ambulances or amount of medicines are limited), he or she performs a top-k query where k is set to 3. If the radio link between the mobile nodes in the bottom-right and bottom-left corners is disconnected, the query-issuing node cannot acquire information on victims collected by the two mobile nodes on the right side. If the mobile node in the bottom-left corner holds a copy of the information held by the two mobile nodes on the right side, the query-issuing node can acquire all the necessary information.

In our previous works, we proposed a query processing method for top-k query in M-P2P networks (Hagihara et al., 2009) that can reduce traffic and keep high accuracy of the top-k result.[1] We also proposed a data replication method for top-k query in M-P2P networks (Hara et al., 2010). In the method proposed in Hara et al. (2010), data items with high scores are frequently replicated on mobile nodes. In this case, if a top-k query is processed by using the method in Hagihara et al. (2009), same data items are sent back to the query-issuing node more than once through long paths because each node does not consider replicas held by others. As a result, the traffic for query processing increases.

In this paper, we propose a query processing method for top-k query considering data replication in M-P2P networks, which is an extension of the method proposed in Hagihara et al. (2009). In this method, each node attaches a query message with the information on its own data items (data item IDs) that have possibility to be included in the top-k result, and thus, each node can know data items held by nodes on the query path. When transmitting a reply, each mobile node does not send back data items held by nodes closer to the query-issuing node than itself. In addition, when a node receives a query message from a node on the path which is different from the path through which the node firstly received a query message for the same query, the node behaves as follows. If the query message contains the same data item IDs as the previously received message and the corresponding data items are held by nodes closer to the query-issuing node than the node, the node does not send back these data items. By doing so, duplicate transmissions of same data items through long paths can be suppressed, and thus, the traffic decreases. Moreover, if a node that received a query message does not have data items that have a possibility to be included in the top-k result, i.e., all of its own data items have lower scores than the minimum score attached in the query message, the node stops transmitting a query message, and starts transmitting a reply message. This is because no change in the information on the candidates of data items included in the top-k result can be seen that all data items included in the top-k result have been already acquired with high probability.

To the best of our knowledge, this is the first work that addresses top-k query processing in M-P2P networks which takes data replication into account. We show that the proposed method can drastically reduce traffic for top-k query processing than other approaches. From this, we can confirm that it is very effective to take data replication into account in top-k query processing.

The remainder of this paper is organized as follows. In Section 2, we present the assumed system model and definitions. In Section 3, we introduce related work, and our previous works proposed in Hagihara et al. (2009) and Hara et al. (2010). In Section 4, we propose a query processing method for top-k query considering data replication in M-P2P networks. In Section 5, we show the results of the simulation experiments. Finally, in Section 6, we summarize this paper.

[1] In our previous works, we used term "mobile ad hoc network (MANET)" instead of "M-P2P network". However, both of them have almost the same meaning, thus, we basically use term "M-P2P" in this paper.

2. System model and definitions

2.1. System model

In this paper, the system environment is assumed to be a M-P2P network in which mobile nodes retrieve data items held by itself and other nodes using a top-k query.

We assign a unique *node identifier* to each mobile node in the system. The set of all mobile nodes in the system is denoted by $M = \{M_1, M_2, \ldots, M_m\}$, where m is the total number of mobile nodes and $M_i (1 \le i \le m)$ is a node identifier. These nodes move freely, thus, the network topology dynamically changes and sometimes network partitioning occurs.

Each data item (or tuple) consists of multiple attribute values. For example, temperature, blood pressure, photograph, and other personal data of each injured person can be a data item in a rescue operation. Another example is a case of sensing multiple observations at a specific location and time, which can be a data item. For simplicity, all data items are assumed to be the same size. We assume two different cases for data update: no update and periodical update (T: data update interval). The latter case often happens in a real situation, e.g., the system automatically performs sensing operations at a specific interval. After a data item is updated by its owner, all its replicas become invalid.

Each node has a storage for replicating at most h data items.

2.2. Definitions

2.2.1. Top-k query

The query-issuing node transmits a query message with the query condition and acquires data items with the k highest scores among all data items held by mobile nodes in the entire network. The query condition designates a kind of data and its attributes, e.g., a kind of data is the information on victims and the attribute is their injured level. The score of a data item is calculated from the query condition and the scoring function. In this paper, we focus on only one query condition, i.e., a specific type of queries. In the case that there are multiple query conditions in the system, our strategy simply can handle them by dividing the storage for replication of each node into the number of query conditions. Here, we do not place any restrictions on scoring functions since our proposed method is independent of them. For example, we can use a linear function and also a similarity function as a scoring function. A linear function calculates the score of data items by weighting values of attributes designated by the query condition. One the other hand, a similarity function calculates the score as the similarity between the attribute values and those (i.e., vector) designated by the user.

2.2.2. Standard score

Standard scores proposed in Hagihara et al. (2009) are N scores of data items, which are attached to a query message. Each standard score $B(i, j)(1 \le j \le N)$, calculated by node M_i, guarantees that at least $(k/N)j \, (1 \le j \le N)$ data items have scores equal to or larger than $B(i, j)$ in the entire network. M_i calculates its standard scores as follows:

$$B(i, j) = S\left(i, \frac{k}{N}j\right) \quad (1 \le j \le N). \tag{1}$$

Here, $S(i, l)$ denotes the lth highest score among the scores calculated by M_i.

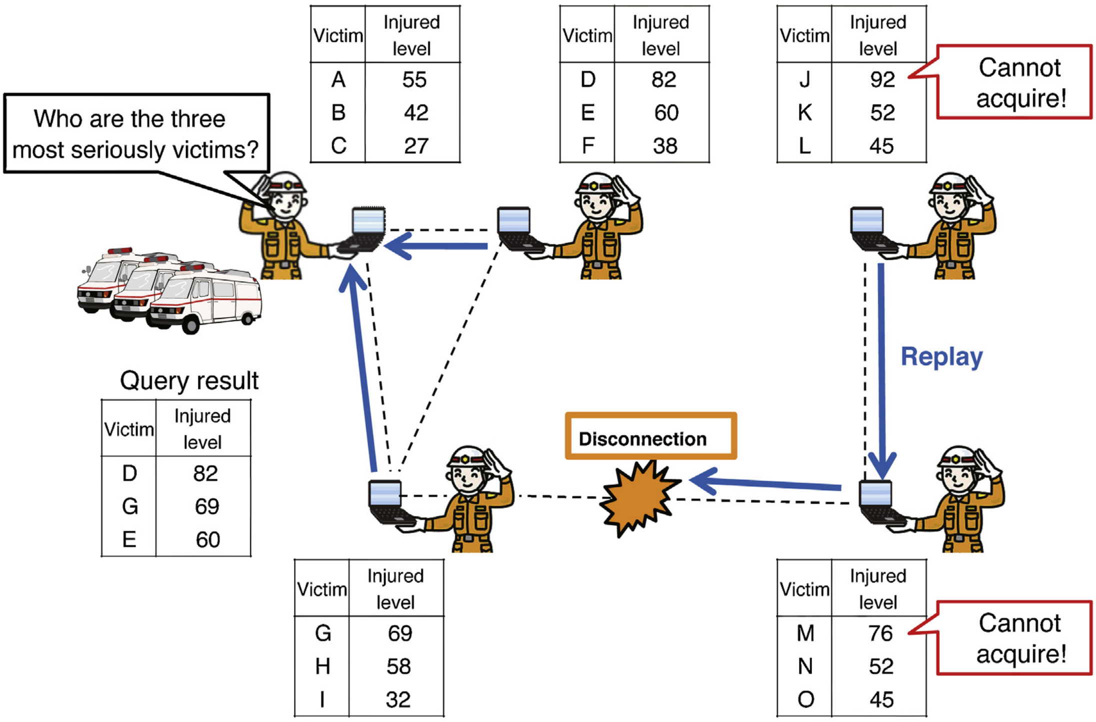


Fig. 1. Example of top-k query in M-P2P networks.

Each mobile node that receives the query message first calculates the standard scores, and then, updates the standard scores as the *N* highest scores among the standard scores in the received queries and those calculated for its own data items. The standard score is useful to reduce the candidate of the top-k result.

3. Related work

In this section, we present some conventional studies on top-k query processing and data replication, which are related to our work. We only focus the three research fields: fixed peer-to-peer (P2P) networks, wireless sensor networks and mobile ad hoc networks (MANETs). They have some similar characteristics such as multihop communication and limited knowledge obtained only from neighbor nodes.

3.1. Top-k query processing

In the research field of fixed P2P networks, many strategies for top-k query processing have been proposed (Akbarinia et al., 2006; Balke et al., 2005; Kalnis et al., 2006; Matsunami et al., 2005; Vlachou et al., 2008). Kalnis et al. (2006) proposed a method in which each node that receives a top-k query message orders its own data items by score, and transmits data items with *k* highest scores. Balke et al. (2005) proposed a method in which a query-issuing node acquires the data item with the highest score by transmitting a query and continues this procedure *k* times to acquire the top-k result. Akbarinia et al. (2006) proposed a method in which each mobile node selectively transmits query and reply messages to reduce network traffic. Matsunami et al. (2005) proposed a method in which each node determines the number of data items included in a reply message based on the number of its children and *k*, and sends back the determined number of data items with highest scores. Vlachou et al. (2008) proposed a method in which super peers collect data items with high scores from its children nodes, and communicate between only super peers. The methods proposed in Balke et al. (2005) and Kalnis et al. (2006) do not work well in M-P2P networks (or MANETs) because these methods respectively cause very large traffic and query processing delay. The method proposed in Matsunami et al. (2005) is not directly applicable to M-P2P networks because each node cannot know the number of its children in query transmissions in advance. The method proposed in Akbarinia et al. (2006) is not applicable to M-P2P networks because it assumes an environment where every message is transmitted by unicast and every two nodes can communicate with each other anytime. The method proposed in Vlachou et al. (2008) can reduce traffic due to limited communication between super peers, but it is difficult to appropriately select super peers in M-P2P networks. Moreover, if a disconnection between super peers occurs, the accuracy of the query result extremely decreases.

In the research field of wireless sensor networks, many strategies for top-k query processing have been proposed (Liu et al., 2010; Malhotra et al., 2011; Ye et al., 2010). Malhotra et al. (2011) proposed a method in which a sink sends the *k* highest score to nodes in order to filter data items that are newly generated but not includes in the top-k result. Only when a node generates data items with scores higher than *k* highest score, it sends the data items to the sink. Ye et al. (2010) proposed a method in which nodes construct a cluster for top-k query processing. Cluster heads send the information on their responsible clusters to the sink, and the sink issues a top-k query based on the information that are sent from cluster heads. Liu et al. (2010) proposed a method in which nodes construct a cluster and cluster heads directly communicate with each other. A query issued from a sink are sent to all nodes through cluster heads, and thus, cluster heads aggregates the data items in their clusters and send the data items to cluster heads closer to the sink. Those methods assume that an only sink collects the data items with *k* highest scores and do not assume the movement of nodes. However, in M-P2P networks which we assume, every node issues a top-k query and moves freely. Therefore, the above-mentioned methods are not suitable for M-P2P networks.

To the best of our knowledge, our previous work (Hagihara et al., 2009) and the method proposed in Padhariva et al. (2011) are only existing studies that address the issue of top-k query processing in

M-P2P networks. However, as described in Section 1, the method in Hagihara et al. (2009) does not assume data replication, and thus, the accuracy of the query result decreases at the point of network partitioning. The method in Padhariva et al. (2011) assumes an economic scheme (e.g., virtual currency) in which the assumption is quite different from our work.

3.2. Data replication

In various research fields, there have been a large number of studies of data replication for improving data availability, response time, fault tolerance, and scalability.

In the research field of MANETs, data replication has been recently studied actively (Cao et al., 2004; Hara, 2001, 2003; Hara and Madria, 2006; Karumanchi et al., 1999; Padmanabhan et al., 2008). In Hara (2001), we proposed three data replication methods in MANETs to improve data availability at the point of network partitioning. In Hara (2003) and Hara and Madria (2006), we extended these methods to adapt to an environment where data updates occur. In Shinohara et al. (2006), we proposed a data access method to replicas considering power consumption in MANETs. This method determines the node that replies to the data request and the path to the query-issuing node made by taking in to account the path length and power consumed by nodes on the path.

In a P2P network, it is common that data items are replicated on multiple peers for efficient data retrieval, improving data availability and load balancing (Cohen and Shenker, 2002; Cuenca-Acuna et al., 2003; Datta et al., 2003). In Cohen and Shenker (2002), the authors discussed the optimal ratios of numbers of replicas allocated in an unstructured P2P network. The authors also discussed that a simple replication algorithm, called *path replication*, can nearly achieve the optimal allocation in terms of query efficiency. In path replication, when a query is successful, the target data item is replicated at all peers along the path from the query-issuing peer to the peer that responded to the query. This method also works well in M-P2P networks because it does not require any extra computational overhead and traffic for data replication.

Most of the conventional studies on data replication in MANET and P2P networks assume that nodes access only a single data item at a time by specifying its data identifier. On the contrary, in this paper, we assume top-k queries where multiple data items are sent to the query-issuing node for each query. Thus, we need to design a different approach for data access (query processing) to improve the system performance.

3.3. Our previous works

In this subsection, we briefly present our previous works; a message processing method for top-k query in Hagihara et al. (2009) and a data replication method for top-k query in Hara et al. (2010).

3.3.1. Message processing method for top-k query

In the method proposed in Hagihara et al. (2009), we aimed at reducing traffic and keeping high accuracy of the query result. In this method, each mobile node estimates the k highest scores of data items and sets N of them to its Standard Scores.

The procedures of the query-issuing node, M_p, and mobile nodes that receive a query message are briefly explained below. First, M_p specifies the number of requested data items k and the query condition. Then, M_p calculates the scores of its own data items from the query condition using the scoring function and initializes its standard scores.

Then, M_p transmits a query message attached with the standard scores to its neighboring mobile nodes. Each mobile node that received the query message first calculates the standard scores for its own data items in the same way as M_p did, and then, updates the standard scores. Then, it forwards the query message attached with the revised standard scores to its neighbors.

The reply messages of the query are sent back to M_p in the reversed routes how the query was propagated. Specifically, if a mobile node, M_s does not have any neighbors (children) except for that already received the query message, it starts sending back a reply message. The reply message includes the threshold and the list of the pair of the scores that are larger than the threshold and the data items with the corresponding scores. The threshold is set as $B(s, N)$ because it guarantees that more than $(k/N)x$ data items with scores equal to or larger than the standard score $B(s, x)$ are in the entire network. When a mobile node, M_t, receives a reply message, it updates the threshold with the higher score between the threshold in the received reply message and $B(t, N)$.

By this method, each mobile node can reduce the number of candidates of data items included in the top-k result, which helps to reduce the traffic in query processing.

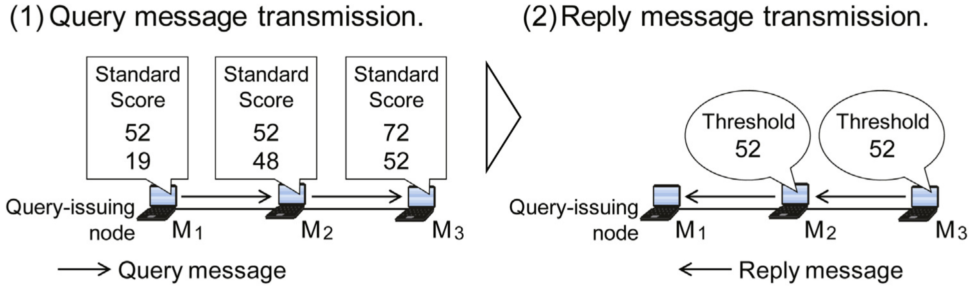
Fig. 2 shows an example when M_1 retrieves data items with the eight highest scores using two standard scores ($k = 8$, $N = 2$). Here, for simplicity, we do not assume any specific query condition and scoring function in this example, but just assume the scores of data items held by each node are given as shown in the left bottom table. The table on the right side shows scores of data items which are sent by each node as the reply for the query (we call this the *reply list*) and scores of data items in the query result. In this table, underlined scores denote data items that are not held by the node. The list in a balloon on the left side denotes two standard scores, which each mobile node sets. The number in a balloon on the right side denotes the threshold set by each node. For example, when M_2 receives a query message from M_1, it compares its own initial standard scores, i.e., 4th and 8th scores of its own data items (48 and 28), with the standard scores (52 and 19) in the query message, sets the two highest scores among them to its own standard scores (52 and 48), and continues to transmit the query message. In the same way, M_3 sets its own standard scores to 72 and 52. Because M_3 does not have any child, it starts to send back a reply attached with its own data items whose scores are equal to or larger than the threshold (52).

3.3.2. Data replication method for top-k query

In the data replication method proposed in Hara et al. (2010), we adopted the path replication proposed in Cohen and Shenker (2002) for P2P networks. In path replication, each node replicates data items when it sends back data items in the reply for a query. This approach is suitable for resource constrained M-P2P networks because it does not cause any extra overhead in terms of traffic and computation. Here, nodes in M-P2P networks are generally have limited storage for data replication, thus, it often happens that nodes cannot replicate all data items in the reply for a top-k query. In particular, as k (the number of requested data items) increases, it is more likely to happen such a situation.

In this case, if every node determines data items to replicate based on the same policy such as replicating data items with higher scores in path replication, many nodes on the path along with the reply message replicate same replicas, i.e., heavy replica duplication occurs. Replica duplication makes kinds of data items replicated among connected nodes fewer, thus, the accuracy of query results at the point of network partitioning becomes lower. Therefore, we designed the proposed replication method so that data items with higher scores are given higher priority and also replica duplication reduces among mobile nodes by adopting a random selection factor.

Moreover, we assumed an environment where periodic data updates occur as well as that without data update. In such an environment, allocating replicas that will become invalid soon is not effective, i.e., those replicas cannot be used for responding to queries for long time. Thus, the proposed method gives higher



Scores of data items held by each mobile node.

Rank \ Node	M_1	M_2	M_3
1	85	77	92
2	70	67	84
3	61	56	80
4	52	48	72
5	44	46	60
6	39	40	59
7	25	32	37
8	19	28	30
...	...	...	...

Reply list from each mobile node and query result.

M_1 Query result	M_2 Reply list	M_3 Reply list
92	**92**	92
85	**84**	84
84	**80**	80
80	77	72
77	**72**	60
72	67	59
70	**60**	
67	**59**	

Fig. 2. Query message transmission for top-k search in the method proposed in Hagihara et al. (2009).

priority to data items with longer remaining time until they are updated next.

4. Proposed method

In this section, we present the query processing method for top-k query proposed in this paper. The basis of our proposed method is same as that of the method in Hagihara et al. (2009). The difference is whether data replication is taken into account. Firstly, we describe the design policy of our proposed method. Then, we describe how each mobile node determines standard scores in our proposed method. We also describe the procedures for transmitting query and reply messages using the standard scores. Finally, we describe the consideration about our proposed method.

4.1. Design policy

As mentioned, to improve data availability at the point of network partitioning in M-P2P networks, the most promising solution is data replication. In data replication for top-k query such as that we proposed in Hara et al. (2010), each node tends to replicate data items with high scores. It means that a large number of same data items with high scores are replicated on mobile nodes in the network. Here, if each node sends back data items (replicas) without considering data items held by other nodes, a large number of same data items are sent back to the query-issuing node, which increases unnecessary traffic. In particular, mobile nodes at the ends (leaf nodes) on query paths tend to start transmitting a large number of data items. These long way transmissions significantly increase traffic. Therefore, we design the proposed query processing method to suppress duplicate and long way transmissions of data items.

In our proposed method, each mobile node estimates the k highest scores of data items and sets N of them to its standard scores in the same way as the method in Hagihara et al. (2009) as described in Section 2.2. However, the method in Hagihara et al. (2009) does not take data replicas into account. In our proposed method, each node calculates the standard scores considering data replicas as well as original data items (primary copies). $S(i, l)$ in Hagihara et al. (2009) denotes the lth highest score among the scores of original data items calculated by M_i. Meanwhile, $S(i, l)$ in our method denotes the lth highest score among scores of original data items and replicas except for the data items included the data ID list (described in the next subsection) calculated by M_i.

In addition, since a large number of mobile nodes have data items (replicas) with high scores, it is expected that all data items included in the top-k result can be acquired from a part of mobile nodes (i.e., not from all nodes) with high probability. Hence, if each node can stop transmitting a query message at a proper timing, unnecessary traffic can be reduced and the search time can be also shortened. We also design our method to achieve this.

4.2. Transmission of query messages

In our method, to reduce the number of candidates of data items that are included in the top-k result, each mobile node attaches a query message with the standard scores. Moreover, to suppress duplicate transmissions of same data items through long paths, each node also attaches a query message with the information on its own data items.

The following is the list of elements included in a query message. A mobile node that sends and receives a query message is simply called a *query sender* and a *query receiver*, respectively.

- Query-issuing node ID
  The node identifier of the query-issuing node.
- Request number
  The number of data items, k, which the query-issuing node requests.
- Query condition
  The query condition specified by the query-issuing node.
- Standard score list
  The list of standard scores determined by the query sender.
- Number of hops

The path length (hop count) from the query-issuing node to the query sender.

- Query sender ID

  The node identifier of the query sender. This is used to recognize the parent of the query receiver.
- Parent ID

  The node identifier of the mobile node that sent the query message to the query sender. This is used to recognize the child of the query receiver.
- Data ID list

  The list of identifiers (IDs) of data items that have a possibility to have the k highest scores among data items held by mobile nodes along the query path from the query-issuing node to the query sender, i.e., the list of IDs of k data items that have a possibility to be included in the top-k result. This is used to recognize the data items held by nodes closer to query-issuing node than the query receiver. The list includes only data item IDs (not scores) in order to minimize the increase of traffic.

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

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

$$B(p,x) = S\left(p, \frac{k}{N}x\right) \quad (1 \le x \le N). \tag{2}$$

Here, $S(i, j)$ denotes the jth highest score among the scores calculated by M_i. Specifically, the standard scores of M_p are the every (k/N)th scores of data items held by M_p.

2 M_p transmits a query message to its neighboring mobile nodes. In the query message, the query-issuing node ID is set as M_p, the request number is set as k, the query condition is set as the specified condition, the standard score list is set as $B(p, x)(1 \le x \le N)$, the number of hops is set as 0, and the data ID list is set as IDs of data items with equal to or higher scores than $B(p, N)$ held by M_p. In addition, the query sender ID is set as M_p, and the parent ID is set as empty.

3 A mobile node, M_q, that receives the query message stores a pair of the number of hops and the query sender ID. If M_q already has received the query message, the procedure goes to step 6. Otherwise, if M_q first receives the query message, it stores the data ID list in the received message. Then, it compares the data ID list with its own data items (including replicas). If M_q has same data items, those data items temporarily become invalid during this query processing time (i.e., M_q behaves as not having those data items). M_q sets the query sender as its parent and calculates the standard scores, $S(q, (k/N)y)(1 \le y \le N)$, for its own data items in the same way as step 1. Then, the procedure goes to step 4.

4 M_q updates the standard scores as follows:

$$B(q,x) = MAX\left(S\left(q, \frac{k}{N}a\right), L(b)\right) \quad (1 \le x \le N). \tag{3}$$

Here, $L(z)(1 \le z \le N)$ denotes the zth highest score listed in the standard score list of the received query message. The initial values of a and b are equal to 1. M_q calculates the next standard score corresponding to x while incrementing a by 1 if $B(q, x)$ is equal to $S(q, (k/N)a)$, i.e., $S(q, (k/N)a) \ge L(b)$, otherwise incrementing b by 1. Specifically, the standard scores of M_q are the N highest scores among $S(q, (k/N)y)(1 \le y \le N)$ and $L(z)(1 \le z \le N)$. Therefore, M_q guarantees that, for each $x(1 \le x \le N)$, there are at least $(k/N)x$ data items with scores equal to or larger than M_q's standard score $B(q, x)(1 \le x \le N)$ among data items held by itself and mobile nodes along the path where the query message was transferred. Then, the procedure goes to step 5.

5 M_q sets the query sender and parent IDs as M_q and M_q's parent, respectively. In addition, M_q updates the standard score list to $B(q, x)(1 \le x \le N)$, increments the number of hops by 1, and updates the data ID list by adding IDs of its own data items with scores equal to or larger than $B(q, N)$ which are not included in the received data ID list. Here, the received data ID list may include IDs of data items with scores smaller than $B(q, N)$, while the scores cannot be known from the received the data ID list. Therefore, if M_q has data items included in the received data ID list, it checks the scores of those data items. If the scores of some of the data items are smaller than $B(q, N)$, M_q deletes the IDs of those data items from the data ID list.

Then, M_q transmits the query message to its neighboring mobile nodes. The procedure goes back to step 3.

6 M_q sets the query sender as its child if the parent ID in the received query message is M_q. Moreover, if M_q is farther from the query-issuing node than the query sender (the distance is based on hop-counts) or it is at the same distance as the query sender and M_q's ID is smaller than the query sender's ID, M_q compares the data ID list in the received query message with its own data items (including replicas). If M_q has same data items, those data items temporary become invalid during this processing query time. By doing so, only node closer to the query-issuing node sends back data items, which can prevent same data items from being sent back from more than one path, i.e., can reduce the traffic. Here, since the sender and receiver nodes are basically within the radio range of each other, each of them can receive messages sent by the other. Thus, both of them can precisely know which one should send back data items held by them, which can prevent same data items being sent back by both nodes, and also can guarantee that one of them surely sends back these data items.

Then, M_q updates its own standard scores $B(q, x)(1 \le x \le N)$ in the same way as step 4 for further reducing the number of candidates of data items included in the top-k result. Here, to avoid the increase of traffic, M_q does not re-transmit the query message including the updated standard scores.

Algorithm 1. Query transmission

```
Require: k and a query condition
Ensure: Data items with k highest scores
    M_p calculates its standard score
    Sets a query message
    Broadcasts the query message
    if Node, M_q receives the query message then
      Stores a pair of number of hops and the query sender ID
      if M_q receives first then
        Stores the data ID list on the query messages
        Makes the data items included in the data ID list temporarily invalid
        parent ← the query sender
        Calculates and updates its standard score
        Sets the query message
        Broadcast the query message
      else if M_q's ID is the parent ID in the query message then
        child ← child ∪ the query sender
      else if M_q's hop count < the query sender's hop count or (same hop counts and M_p's ID < the query sender's ID) then
        Stores the data ID list on the query message
        Makes the data items included in the data ID list temporarily invalid
        Updates its standard score
      end if
    end if
```

Algorithm 1 shows the pseudocode for the query transmission procedure. In this algorithm, M_p and M_q respectively denote the query-issuing node and nodes that received query messages. The procedure of the query-issuing node (steps 1 and 2) is described in lines 1–3. Then, the procedure of nodes that received the query message first time (steps 3–5) is described in lines 4–12, and the procedure of nodes that received the query message again (step 6) is described in lines 13–19.

In the transmission of a query message, each mobile node can reduce the candidates of data items that are included in the top-k result by using standard scores. Moreover, each node can know data items held by the nodes closer to the query-issuing node than itself, which is helpful to further reduce the candidate of data items included in the top-k result.

By specifying the query sender and parent IDs in a query message, each mobile node can recognize its parent and children in the query transmission tree whose root is the query-issuing node. Each mobile node also can recognize the relationship between itself and the neighboring nodes which are neither its parent nor children by using pairs of the number of hops and the query sender ID.

Fig. 3 shows an example when M_1 retrieves data items with the six highest scores using two standard scores ($k=6$, $N=2$). In this figure, the table shows scores of original data items and replicas held by each node. In this table, highlighted scores are data items included in a data ID list in a received query message (i.e., the data items temporarily become invalid). The list in a balloon denotes two standard scores, which each mobile node sets. For example, M_1 sets 87 and 61, i.e., 3rd and 6th scores among its original data items and replicas, as its own standard scores, and adds identifiers of data items with equal to or higher scores than 61 to its data ID list. Then, M_1 transmits the query message to neighboring nodes. M_2 that received the query message knows the data items which are not necessary to reply (96, 87 and 84) because the identifiers of these data items are included in the data ID list in the query message. Additionally, M_2 compares its own initial standard scores, i.e., 3rd and 6th scores of its data items except for the invalid one (74 and 30), with the standard scores in the query message (87 and 61), and sets the two highest scores among them to its own standard scores (87 and 74). Then, M_2 continues to transmit the query message after adding identifiers of data items with equal to or higher scores than 74 to the data ID list. M_3 and M_4 that received the query message from M_2 do procedures in the same way as M_2. Moreover, M_3 receives the query message from M_4 again after receiving the query message from M_2. In this case, as the hop counts of M_3 and M_4 from M_1 are same and the identifier of M_3 is smaller than that of M_4, M_3 makes the data items included in the data ID list (80) invalid. On the other hand, although M_4 receives the query message from M_3, M_4 does nothing because the identifier of M_4 is larger than that of M_3.

4.3. Transmission of reply messages

Since each mobile node can reduce the candidates of data items included in the top-k result as described in Section 4.2, it transmits a reply message attached with only those candidates. In addition, if a mobile node that received reply messages from other mobile nodes finds some data items unnecessary, i.e., surely not included in the query result from the received information, it does not transmit those data items.

The following is the list of elements included in a reply message.

- Query-issuing node ID
  The node identifier of the query-issuing node.
- Threshold
  The kth highest score estimated by the mobile node that transmits the reply message.
- Reply list
  The list of data items (and their scores) with scores equal to or larger than the threshold that are not included in the data ID list received during the query transmission phase.

The procedure of each mobile node that has finished the transmission of the query message is as follows.

1 Each mobile node, M_s, which has no children, starts transmitting a reply message to its parent. In the reply message, the query-issuing node ID is set as M_p. The threshold is set as $B(s, N)$ because it is guaranteed that at least k data items with scores equal to or larger than $B(s, N)$ exist in the entire network. The reply list contains the list of data items with scores equal to or larger than the threshold $B(s, N)$ that are not included in the data ID list received during the query transmission phase.
2 Each mobile node, M_t, that received a reply message updates the threshold as the higher score between the threshold in the received reply message and $B(t, N)$. M_t deletes data items with scores smaller than the threshold from the reply list in the received reply message. Instead, M_t inserts its own data items with scores equal to or larger than the threshold to the reply list. Then, if the number of data items contained in the reply list becomes larger than k, M_t updates the threshold with the kth score in the reply list and deletes scores, which are smaller than the new threshold, and data items with the corresponding scores in the reply list.
  When M_t receives reply messages from all its children, it transmits the updated reply message to its parent.
  This procedure repeats until the query-issuing node receives the reply message from its all children.

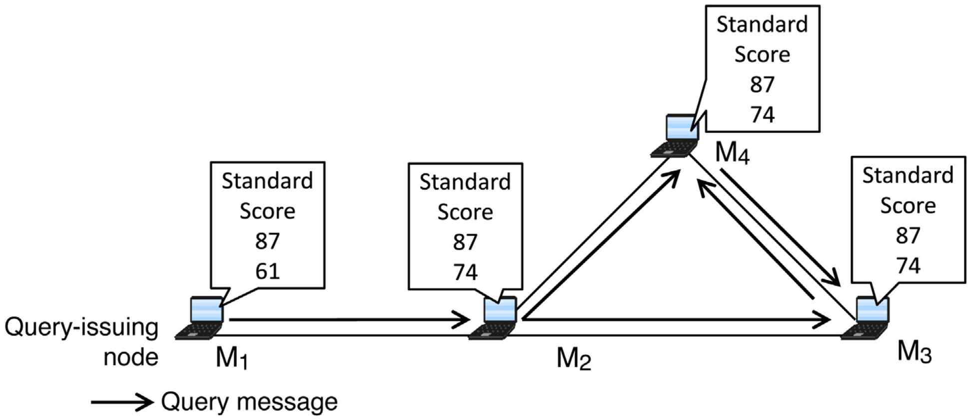
In our method, each mobile node does not send data items that are not included in the top-k result using the threshold, which can reduce traffic for processing a top-k query. In addition, since each node does not send data items that are included in the data ID list, duplicate transmissions of same data items through long paths can be suppressed, which is also effective to reduce traffic.

4.4. Detection of radio link disconnection

In M-P2P networks, the network topology dynamically changes due to the movement of mobile nodes. If a mobile node cannot transmit a reply message to its parent because of a disconnection of the radio link between these nodes, the accuracy of the query result decreases. Therefore, when a mobile node detects a disconnection of the radio link to its parent, the mobile node searches for an alternative path through other nodes to transmit the reply message along the discovered path. Nodes that receive the path-search message can be differentiated between reply and path-search messages from the elements included in the message.

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

1 When M_u detects a disconnection to its parent, it transmits a reply message to a neighboring mobile node except for M_u's children. If there are multiple candidates, M_u selects the mobile node with the smallest number of hops to reduce traffic and delay as much as possible.
2 When a mobile node, M_v, receives a reply message from a mobile node which is not M_v's child, it transmits a reply message to its parent as follows. If M_v is waiting for reply messages from its children, M_v treats the received reply message from the (non-child) node in the same way as M_v's children. Otherwise, if M_v has already transmitted the reply message, M_v updates the threshold and transmits the new reply message to its parent in the same



Scores of original data items and replicas held by each mobile nodes

Rank \ Node	M1		M2		M3		M4	
	Original	Replica	Original	Replica	Original	Replica	Original	Replica
1	90	96	96	92	92	90	81	90
2	75	87	87	84	84	72	72	80
3	61	84	74	81	80		68	
4	53		63		78		60	
5	48		45		65		56	
6	42		30		59		43	
...	...		...		...		...	

Fig. 3. Query message transmission for top-k search.

way as described in Section 4.3. Here, to prevent M_v from redundantly re-transmitting the same data items, M_v deletes the data items that were already transmitted or included in the data ID list from the reply list in the new reply message.

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

- Sender node ID
  The node identifier of the mobile node that transmits the path-search message.
- Path list
  The list of node identifiers of mobile nodes along the path where the path-search message is transmitted.

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

1 M_u transmits a path-search message to all its children. In this message, the sender node ID is set as M_u and the path list is set as only M_u.
2 When a mobile node, M_v, receives a path-search message, it checks whether a neighboring mobile node, whose parent is not the sender node of the received message, exists or not. If such a mobile node exists, it is obvious that the path from the node to the query-issuing node does not include the disconnected radio link. Thus, M_v adds the path information from M_v to the query-issuing node via the found node to the path list in the message received from M_u and sends it back to M_u. When M_u receives the path list from M_v, it transmits the reply message along the newly discovered path, and M_v processes the reply message in the same way as described in Section 4.3.
  Otherwise, if such a mobile node does not exist, M_v inserts itself into the path list in the received path-search message and transmits the message to its children.

In the procedure of handling a radio link disconnection, each mobile node transmits a reply message through the discovered path even if the radio link to its parent is disconnected. Therefore, the query-issuing node receives all necessary reply messages with high probability.

4.5. Query stop

The top-k result may be retrieved from a small part of nodes near the query issuing node due to replicas. Hence, if the list of data items that have a possibility to be included in the top-k result is not updated at a mobile node that received a query message, the node stops transmitting the query message, i.e., it judges that all data items included in the top-k result can be already acquired from mobile nodes on the query path before itself. We add the following procedure to the end of step 5 in the query transmission described in Section 4.2.

- Otherwise, if the data ID list has not changed, i.e., M_q does not have data items that are added to the data ID list, M_q stops transmitting the query message (which we refer to "*Query Stop (QS)*"), and starts transmitting the reply message. The procedure finishes.

Each node can stop transmitting a query message at a proper timing using QS, and thus, the traffic and the search time can be reduced. There is a trade-off relationship between "traffic and search time" and "accuracy of query result". QS is adopted to balance this trade-off, but it may not work well if it stops query transmissions at improper timing. From our preliminary experiments and analysis, we confirmed that the probability that QS stops query transmission at improper timing is basically very low. For example, when $h = 100$ (the number of replicas allocated at each node) and $k = 100$, if the data ID list already includes 99 and 98 data items, the probability that one and two data items that should be included in the top-k result are missing due to improper timing of QS are respectively about 20% and 3.9%. Also, when "$h = 100$ and $k = 1$", "$h = 20$ and $k = 100$", "$h = 20$ and $k = 20$", and "$h = 20$ and $k = 1$", the probabilities were about "1.1% and –", "42.1% and 17.7%", "38.7% and 14.7%", and "28.1% and –", respectively. Here, while these probabilities are relatively high, "$h = 20$ and $k = 100$" and "$h = 20$ and $k = 20$", QS rarely occurs. This is because in most cases, queries are propagated to leaf nodes in query transmissions before collection an enough number of data items to activate QS. Even if some data items are missing on a query path, these data items are replied from other query paths with high probability. Actually, we confirm in our simulation that the accuracy of query result rarely decreases due to QS.

4.6. Discussion

Our proposed method does not reply data items held by nodes closer to the query-issuing node than itself for reducing the number of replied data items, and thus, the traffic can be reduced. To achieve this purpose, the query message is attached with the information on what data items the nodes on the query path hold. As a result, the size of a query message increases. However, since the size of a data item is generally much larger than the size of a query message, it is effective to reduce the number of replied data items. If there are no same replicas held by nodes on the query path, the message size in our proposed method becomes large uselessly. However, since multiple nodes tend to hold same replicas with high scores with high probability, our proposed method is expected to work well.

It is also expected that the accuracy of query result does not decrease compared with the previous method. Of course, if QS stops query transmissions at improper timing, the accuracy of query result may decrease. However, as described above, it rarely occurs and even if it does occur, missing data items can be replied from other query paths with high probability. Here, there is another approach to reduce replied data items (traffic) and query processing delay, where nodes frequently exchange the information of data items held by themselves to know the distribution of replicas in advance. However, this approach is not suitable for M-P2P networks because of limited network bandwidth and dynamic changes of network topology and data values. Since our proposed method does not exchange any messages in advance, it does not require any maintenance costs and can acquire the necessary information only during the query propagation.

5. Simulations

In this section, we show results of simulation experiments regarding performance evaluation of our proposed method. For the simulation experiments, we used the network simulator, QualNet4.0 (Scalable Network Technologies).

5.1. Simulation model

Mobile nodes exist in an area 600 m × 600 m. The initial position of each node is determined randomly. The number of mobile nodes in the entire network is 50. These nodes move based on the random waypoint model (Camp et al., 2002), where the movement speed and the pause time are s [m/s] and 60 s. Each mobile node transmits messages (and data items) using an IEEE 802.11b device whose data transmission rate is 11 Mbps. The transmission power of each mobile node is determined so that the radio communication range becomes about 100 m. Packet losses and delays occur due to radio interference.

Table 1
Parameter configuration.

Symbol	Meaning	Values (range)
s	Movement speed	3.0 (0–5.0)
h	Number of allocated replicas	50 (0–100)
k	Number of requested data items	50 (20–100)
N	Number of standard scores	10 (5–50)

Each mobile node holds 100 data items. The size of data items is 1 kB. The score of each data item is randomly determined within the range between 0 and 100. We assume both cases where data updates do not occur and do periodically occur at interval 2000 s. In the latter case, the time when each data item is updated at the first time is randomly determined within the range between 0 and 2000 s. Each mobile node can replicate h data items on its storage.

Every mobile node issues a top-k query which the number of requested data items is set as k in rotation at time interval of 60 s. We assume that top-k queries are processed by using our proposed method, the method proposed in Hagihara et al. (2009) (which we call the "previous method"), where the number of standard scores is set as N in both methods, the method proposed in Matsunami et al. (2005) (which we call the "method for fixed P2P") and the method proposed in Balke et al. (2005) (which we call the "naive method"). In the method for fixed P2P, the number of data items included in a reply message is determined by the equation, $k_i = (k_{i-1} \times r_m)/(N_{p_{i-1}})$. In this equation, k_i is the number of data items included in its own reply message (i denotes the hopcount from the query-issuing node), k_{i-1} is that in its parent's reply message ($k_0 = k$), r_m is a positive constant for margin, and $N_{p_{i-1}}$ is the number of its parent's children. This equation means that the number of data items in a reply message becomes smaller for nodes with longer hops from the query-issuing node. The query-issuing node can acquire k data items when $r_m = 1$. Here, the problem to adopt this method in M-P2P network is that each node does not know in advance the number of its children in query transmissions. Thus, we simply use $N_{p_{i-1}} = 2$ for all nodes. When $N_{p_{i-1}} = 1$, this method is identical to the method proposed in Balke et al. (2005). In the naive method (proposed in Balke et al., 2005), each node replies k data items with highest scores held by itself and its descendants. Therefore, the query-issuing node can completely acquire the top-k result when network partitioning and packet loss do not occur, while the traffic becomes very large. Since the two methods proposed in Balke et al. (2005) and Matsunami et al. (2005) assume fixed unstructured P2P (not M-P2P) networks, they do not take nodes' mobility and link disconnection into account. Thus, the same procedure to handle link disconnection as that of our method is applied to these two methods.[2] In all of the four methods, the data replication method proposed in Hara et al. (2010) is used for data replication.

Table 1 summarizes the parameters used in our simulations. The parameters are basically fixed to constant values specified by numbers to the left of the parenthetic values. Each of them is varied in the range specified by the parenthetic element in a simulation experiment. In the above simulation model, we evaluate the following criteria. These criteria are defined as the average for all 50 queries issued during the simulation time.

[2] We examined the effects of handling link disconnection in Hagihara et al. (2009).

Table 2
Parameter configuration.

Type	Method	Size [B]
Query	Our method	$200+4j$
	Previous method	196
	Naive method and method for fixed P2P	36
Reply	Our method and previous method	$26+(1016)i$
	Naive method and method for fixed P2P	$18+(1016)i$
Path-search	All methods	$2+4l$

1 *Accuracy of query result* is defined as the ratio of the numbers of data items acquired by the query-issuing node that are included in the top-k result to the number of requested data items, k.
2 *Traffic* is defined as the total volume of query messages and replies exchanged for processing a query. Table 2 shows the size of each message used in our methods and the method proposed in Hagihara et al. (2009). In this table i denotes the number of data items whose information is included in the reply, j denotes the number of IDs of data items included in the data ID list, and l denotes the number of node identifiers of mobile nodes included in the path list in the path-search message.
3 *Search time* is defined as the elapsed time after the query-issuing node issued a top-k query until it acquired the result.

Here, we discuss the properness of our simulation setting. First, regarding the communication, we assume 802.11.b and consider the effects of underlying network layers and physical factors (e.g., packet loss and communication delay), which is close to a real situation. Next, the area where mobile nodes exist is a plane area (no obstacle) and its size is 600 m × 600 m. In the research field of M-P2P network (or MANET), the area is usually assumed as a plane area for performance evaluation, and thus, we also adopt it. Here, our proposed method can work regardless of the shape of the area. The data size is same for all data items for simplicity. This situation often exists in a real situation. For example, when a relational database is used for data management, tuples are generally composed of a set of elements whose sizes are fixed, and thus, the tuple (data item) size is also fixed. Furthermore, even if the data size is different among data items, our proposed method can work without any problem. Regarding the mobility model, we assume a situation where each member such as rescue worker works individually, and thus, we use the random waypoint model, which is the most popular mobility model. We assume that members move in a walking or running speed. Regarding the maximum number of replicas allocated at each node (h), since there are various kinds of query (queries with different scoring functions) in a real situation, we assume that the limited number of replicas, h, can be allocated for each kind of query, which is not very large. In our simulations, for simplicity, we assume only one scoring function, i.e., one kind of query. In a real situation, when there are several kinds of query, the number of allocated replicas for each kind of query should be appropriately determined according to the characteristics of queries such as the query-issuing rate and the average value of k. In our simulations, we change the value of h in a wide range to fully examine the impact of h (which also means the change of the ratio of h to k).

5.2. Simulation results in an environment without data update

First, we show the results of simulations in an environment without data update.

5.2.1. Impact of number of allocated replicas

We examine the effects of h, the maximum number of replicas allocated on each node. Fig. 4 shows the simulation result. In these graphs, the horizontal axis indicates h. The vertical axes indicate the accuracy of query result in the case of (a), the traffic in the case of (b), and the search time in the case of (c). In these graphs, "Proposed" denotes our proposed method without QS (Query Stop), "Proposed with QS" denotes our proposed method with QS, "Previous method" denotes the method proposed in Hagihara et al. (2009), "Method for fixed P2P" denotes the method proposed in Matsunami et al. (2005), and "Naive" denotes the method proposed in Balke et al. (2005).

From Fig. 4(a), when $h=0$ (no replicas), the accuracy of query result is about 70% in all methods. As h increases, the accuracy of query result increases in all methods. This is because accessible data items increases due to the allocation of replicas. The accuracy of query result in our proposed method is higher than that in other methods for every h. This is because the number of data items included in a reply message is small in our proposed method, and thus, the traffic is also low, which causes fewer packet losses than other methods. In the previous method, as h increases, the number of replied data items increases, and the traffic increases. Therefore, packet losses occur more often than our proposed method. In the method for fixed P2P, the number of replied data items is small, i.e., the traffic is low, while the accuracy of query result is low. This is because multiple nodes tend to reply same data items when replicas are allocated. The accuracy of query result in the naive method is almost same as that in the previous method.

From Fig. 4(b), the traffic in our proposed method decreases as the number of allocated replicas increases. On the other hand, on the previous method, the traffic increases because the number of replied data items increases. In the method for fixed P2P and the naive method, the traffic is constant because the number of replied data item is determined independently of the number of allocated replicas.

From Fig. 4(c), we can see that the search time has a correlation with traffic. This is because as the traffic increases, the transmission delay increases, and thus, the search time also increase. However, in our proposed method (without QS), as h increases, the traffic decreases, but the search time is almost constant. This fact shows that the search time does not change when the traffic is less than 1500 kB. In our proposed method with QS, the search time decreases as h increases. This is because QS occurs more frequently, and the total hopcount for query propagation becomes smaller.

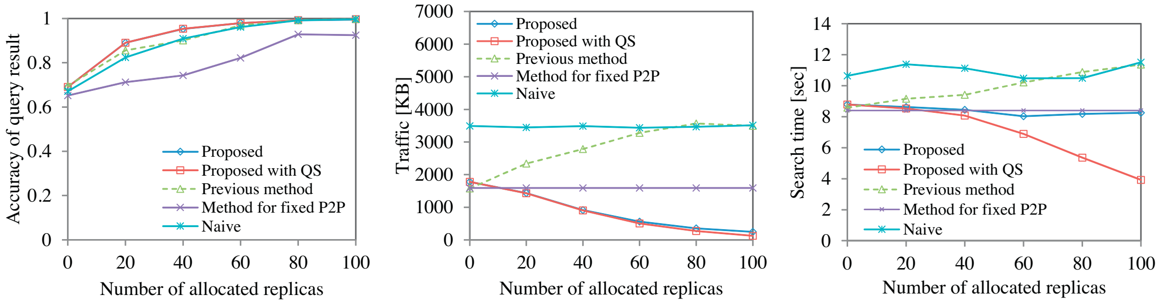
5.2.2. Impact of movement speed

We next examine the effects of s, the movement speed of each node. Fig. 5 shows the simulation result. In these graphs, the horizontal axis indicates s. The vertical axes indicate the accuracy of query result in the case of (a), the traffic in the case of (b), and the search time in the case of (c).

From Fig. 5(a), when $s=0$ (mobile nodes do not move), the accuracy of query result is low in all methods. This is because the network topology does not change, and thus, some data items that are inaccessible due to network partitioning (in the initial state) cannot be acquired during the entire simulation time. When nodes move, replicas can be distributed to nodes in a wide range due to network topology change, and thus, the accuracy of query result increases even if network partitioning occurs. The accuracy of query result in our proposed method is higher than that in other methods because the traffic is lower and packet losses less occur.

From Fig. 5(b), the traffic in our proposed method is smaller than that in other methods. This is because the number of replied data items is smaller as described above. When $s=1$ and $s=2$, the traffic in all methods is the higher than that in other cases. When $s=0$, the traffic is low because the number of available data items is low. As the movement speed increases, since a link disconnection during query processing frequently occurs, the number of replied data items decreases, and thus, the traffic decreases.

From Fig. 5(c), the search time in our proposed method and that in the method for fixed P2P are less than that in other methods.



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

Fig. 4. Impact of *h* in an environment without data update.

This is due to the same reason as that in Fig. 4(c). When $s = 0$, since link disconnections do not occur and each node can receive replies from all its children with high probability, the search time is short.

5.2.3. *Impact of number of requested data items*

We next examine the effects of *k*, the number of requested data items. Fig. 6 shows the simulation result. In these graphs, the horizontal axis indicates *k*. The vertical axes indicate the accuracy of query result in the cased of (a), the traffic in the case of (b), and the search time in the case of (c).

From Fig. 6(a), when *k* is small, the accuracy of query result is very high in all methods except for the method for fixed P2P. This is because each node has the exact top-k result in its storage. The accuracy of query result in the method for fixed P2P is lower than that in other methods because the number of replied data items is small and thus some replicas included in the top-k result are often missing. As *k* increases, the accuracy of query result decreases because the query-issuing node cannot acquire the top-k result from its own replicas and packet losses occur during collecting data items held by other nodes.

From Fig. 6(b), as *k* increases, the traffic in all methods increases. This is because the number of replied data items increases. When *k* is smaller than the number of allocated replicas, the traffic in our proposed methods is almost zero, because the query-issuing nodes can acquire the top-k result from its own replicas in most cases.

From Fig. 6(c), as *k* increases, the search time basically increases in all methods because the number of replied data items increases. In the proposed method with QS, when *k* is small, the search time is significantly smaller than that in other methods. This is because it stops transmitting a query message soon, while other methods flood a query message into the network.

5.2.4. *Impact of number of standard scores*

We next examine the effects of *N*, the number of requested data items. From Fig. 7, the performance of each method does not much change even if *N* changes. However, the traffic of each method first decreases and then increases as *N* increases. This is because, when *N* is small, the size of a query message is small, but the size of a reply message is very large. When *N* is very large, the size of a query message unnecessarily is large, while the size of a reply message does not change. Thus, an appropriate number of standard scores *N*, depends on several factors such as *k*, the number of nodes, and data size. The search time in the proposed method with QS slightly decreases as *N* increases. This is because the candidates of replied data items are narrowed down, and thus, a query stop occurs more frequently.

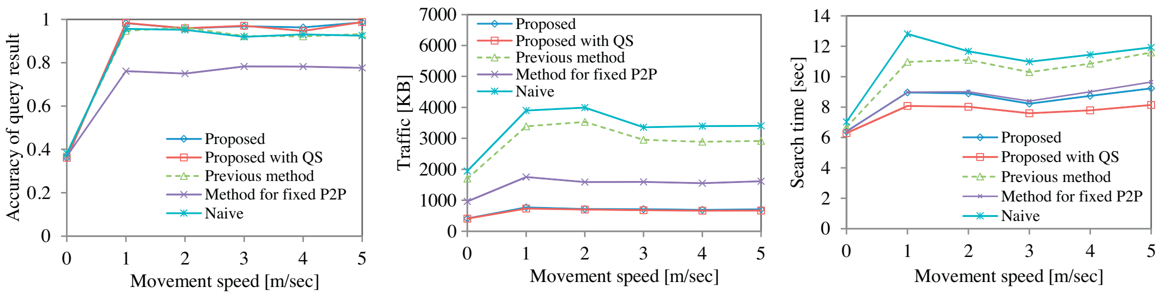
5.3. *Simulation results in an environment with data update*

Next, we show the results of simulations in an environment with periodic data update.

5.3.1. *Impact of number of allocated replicas*

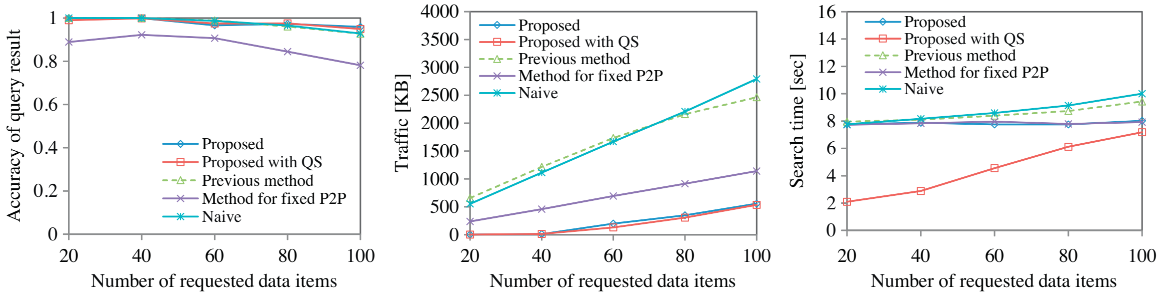
We examine the effects of *h*, the maximum number of replicas allocated on each node in a environment where periodic data updates occur. Fig. 8 shows the simulation result. In these graphs, the horizontal axis indicates *h*. The vertical axes indicate the accuracy of query result in the case of (a), the traffic in the case of (b), and the search time in the case of (c).

Fig. 8(a) and (b) shows the similar characteristics as that of the case without data update. Compared with the result in Fig. 4(a), the accuracy of query result decreases because replicas become invalid after their original data items are updated and those invalid data replicas cannot be used for responding to top-k queries. The traffic in our proposed method increases compared with the result in Fig. 4(a), while that in the previous method decreases. This is because our proposed method can reduce the number of data items in reply messages more effectively when there exist more valid replicas. On the other hand, the method in Hagihara et al. (2009) produces more traffic when there exist more valid replicas, because



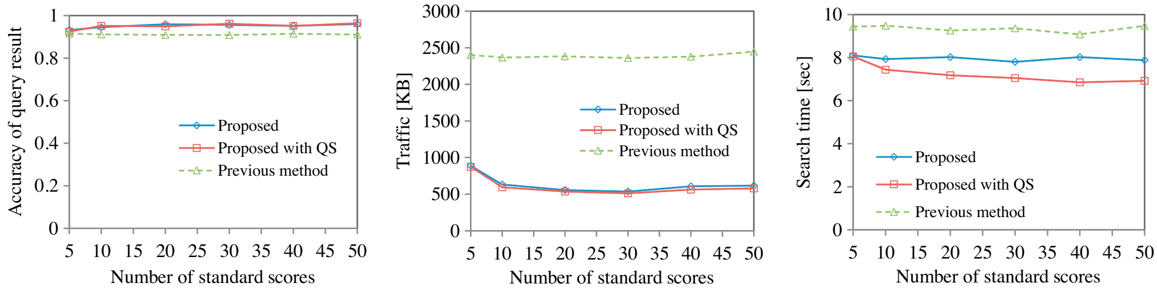
(a) Accuracy of query result (b) Traffic (c) Search time

Fig. 5. Impact of *s* in an environment without data update.



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

Fig. 6. Impact of k in an environment without data update.



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

Fig. 7. Impact of N in an environment without data update.

of duplicated transmissions of data items along long paths. In the method for fixed P2P, the accuracy of query result increases compared with the result in Fig. 4(a). This is because when updates occur, duplicated replicas become fewer, and the replica duplication is the main reason of the degradation of accuracy in this method.

From Fig. 8(c), the search time in our proposed method basically increases compared with the result in Fig. 4(c). This is because the number of valid replicas decreases. In particular, the mechanism of stopping the query transmission does not effectively work.

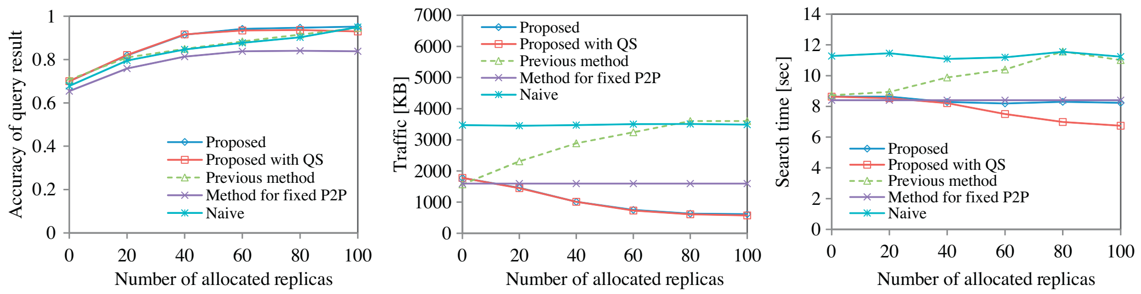
5.3.2. Impact of movement speed

We next examine the effects of s, the movement speed of each node in an environment where periodic data updates occur. Fig. 9 shows the simulation result. In these graphs, the horizontal axis indicates s. The vertical axes indicate the accuracy of query result in the case of (a), the traffic in the case of (b), and the search time in the case of (c). Fig. 9 shows the same characteristic as that in Fig. 5 (without data update).

5.3.3. Impact of number of requested data items

We next examine the effects of k, the number of requested data items in an environment where periodic data updates occur. Fig. 10 shows the simulation result in an environments with periodic data update. In these graphs, the horizontal axis indicates k. The vertical axes indicate the accuracy of query result in the cased of (a), the traffic in the case of (b), and the search time in the case of (c).

From Fig. 10(a), as k increases, the accuracy of query result basically decreases except for the proposed method with QS and the method for fixed P2P because of same reason as mentioned above. When k is small, the accuracy of query result in the proposed with QS and the method for fixed P2P is low. This is because the proposed method with QS often stops transmitting query messages at an improper timing, and nodes in the method for fixed P2P tends to redundantly send same data items. As k increases, the accuracy of query result increases in the proposed method with QS, because it more often stops transmitting query messages at a proper timing, which lastly achieves almost same accuracy as the proposed method without QS. When k is large, our proposed methods achieve higher accuracy of query result than other methods because of fewer packet losses.



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

Fig. 8. Impact of h in an environment with periodic data update.

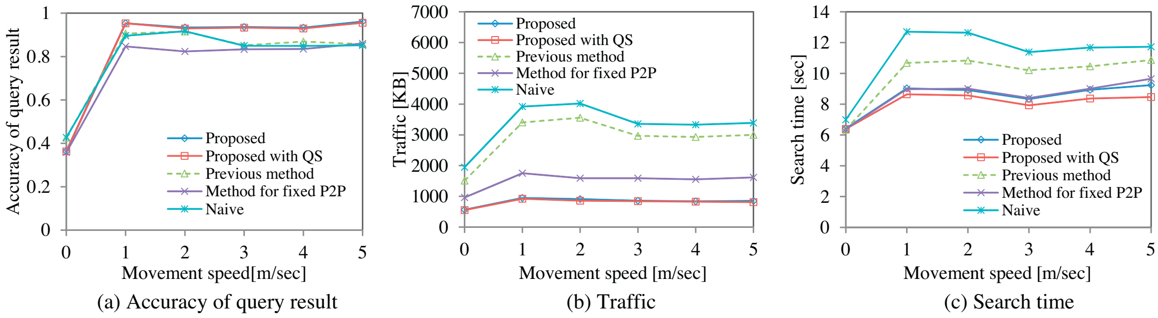


Fig. 9. Impact of *s* in an environment with periodic data update.

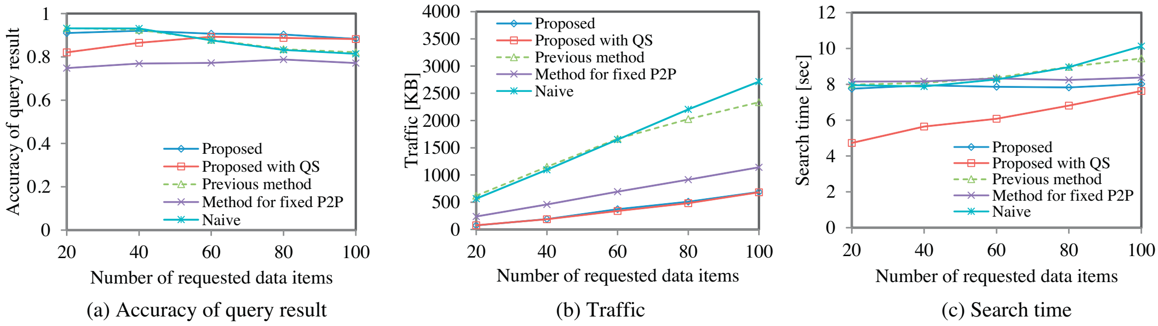


Fig. 10. Impact of *k* in an environment with periodic data update.

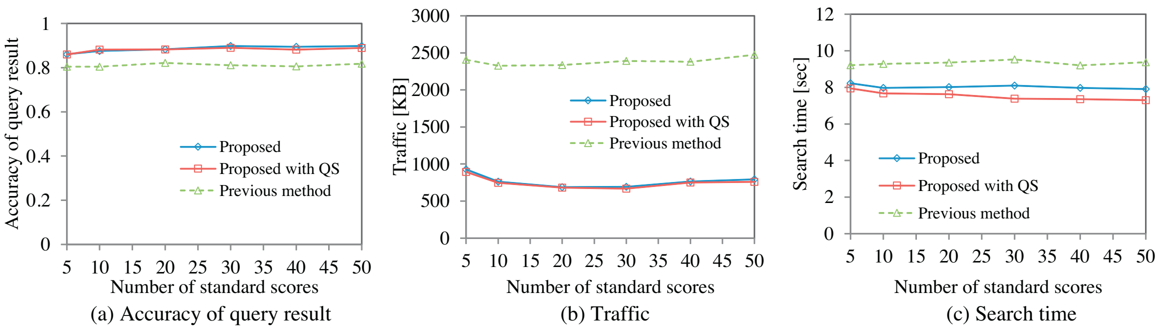


Fig. 11. Impact of *N* in an environment with periodic data update.

Fig. 10(b) and (c) basically show the same features as that in Fig. 6(b) and (c) (without data update). When *k* is small, the search time in the proposed method with QS is larger than that in Fig. 6(c). The is because replicas in an environments with data update have more diversity than that for replica selection without data update, because each node takes into account the data update time, resulting in less occasion of query stop.

5.3.4. *Impact of number of standard scores*

We next examine the effects of *N*, the number of standard scores in an environment where periodic data updates occur. Fig. 11 shows the simulation result in an environments with periodic data update. In these graphs, the horizontal axis indicates *N*. The vertical axes indicate the accuracy of query result in the cased of (a), the traffic in the case of (b), and the search time in the case of (c). Fig. 11 shows the same features as that in Fig. 7.

6. Conclusion

In this paper, we proposed a query processing method for top-k queries considering data replication in M-P2P networks. The proposed method aims at keeping high accuracy of the top-k result and reducing traffic. For this aim, each node attaches a query message with the information on its own data items, and thus, duplicate transmissions of same data items through long paths can be suppressed. In addition, each node stops transmitting a query message at an appropriate timing, and thus, the search time can be reduced.

The simulation results showed that our proposed method can achieve high accuracy of the top-k result and can reduce the traffic and search time.

As the simulation results showed, in our proposed method, the opportunity of replicating data items is fewer than our previous method proposed in Hagihara et al. (2009). Thus, as part of our future work, we plan to extend our proposed method to achieve more effective replica allocation as well as reduce the traffic for query processing. Moreover, we plan to address the issue of deciding whether QS is used or not according to the situation, e.g., *k*, *h*, and skew in replica allocation.

Acknowledgments

This research is partially supported by the Grant-in-Aid for Scientific Research (S)(21220002), (B)(24300037), and JSPS Fellows

(24-293) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

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

Balke, W.-T., Nejdl, W., Siberski, W., Thaden, U., 2005. Progressive distributed top-k retrieval in peer-to-peer networks. In: Proc. ICDE'05, pp. 174–185.

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

Cao, G., Yin, L., Das, C.R., 2004. Cooperative cache-based data access in ad hoc networks. IEEE Computer 37 (2), 32–39.

Cohen, E., Shenker, S., 2002. Replication strategies in unstructured peer-to-peer networks. In: Proc. SIGCOMM'02, pp. 177–190.

Cuenca-Acuna, F.M., Martin, R.P., Nguyen, T.D., 2003. Autonomous replication for high availability in unstructured P2P systems. In: Proc. SRDS, pp. 99–108.

Datta, A., Hauswirth, M., Aberer, K., 2003. Updates in highly unreliable, replicated peer-to-peer systems. In: Proc. ICDCS'03, pp. 76–85.

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: Proc. MDM, pp. 11–20.

Hara, T., 2001. Effective replica allocation in ad hoc networks for improving data accessibility. In: Proc. IEEE Infocom, pp. 1568–1576.

Hara, T., 2003. Replica allocation methods in ad hoc networks with data update. ACM–Kluwer Journal on Mobile Networks and Applications 8 (4), 343–354.

Hara, T., 2010. Quantifying impact of mobility on data availability in mobile ad hoc networks. IEEE Transactions on Mobile Computing 9 (2), 241–258.

Hara, T., Hagihara, R., Nishio, S., 2010. Data replication for top-k query processing in mobile wireless sensor networks. In: Proc. SUTC, pp. 115–122.

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

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

Karumanchi, G., Muralidharan, S., Prakash, R., 1999. Information dissemination in partitionable mobile ad hoc networks. In: Proc. SRDS'99, pp. 4–13.

Liu, X., Xu, J., Lee, W.-C., 2010. A cross pruning framework for top-k data collection in wireless sensor networks. In: Proc. MDM, pp. 157–168.

Malhotra, B., Nacimento, M.-A., Nikolaidis, I., 2011. Exact top-k queries in wireless sensor networks. IEEE Transactions Knowledge and Data Engineering 23 (10), 1513–1525.

Matsunami, H., Terada, T., Nishio, S., 2005. A query processing mechanism for top-k query in P2P networks. In: Proc. SWOD, pp. 84–87.

Padhariva, N., Mondal, A., Goval, V., Shankar, R., Madria, S.-K., 2011. EcoTop: an economic model for dynamic processing of top-k queries in mobile-P2P networks. In: Proc. DASFAA, pp. 251–265.

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

Scalable Network Technologies: "QualNet", http://www.scalable-networks.com

Shinohara, M., Hayashi, H., Hara, T., Nishio, S., 2006. A data access method considering power consumption in ad hoc networks. In: Proc. ISWPC, pp. 193–198.

Vlachou, A., Doulkeridis, C., Norvag, K., Vazirgiannis, M., 2008. On efficient top-k query processing in highly distributed environments. In: Proc. SIGMOD, pp. 753–764.

Ye, M., Liu, X., Lee, W.-C., Lee, D.-L., 2010. Probabilistic top-k query processing in distributed sensor networks. In: Proc. IEEE Int. Conf. on Data Engineering, pp. 585–588.

Yuya Sasaki received the B.E. degree in the Multimedia Engineering and the M.E. degree in the Information Science and Technology from Osaka University, Osaka, Japan, in 2009 and 2011, respectively. Currently, he is a Ph.D. candidate in the Information Science and Technology from Osaka University, Osaka, Japan. His research interests include data search and replication mechanisms in mobile computing environments.

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 an Associate Professor of the Department of Multimedia Engineering, Osaka University. He has published more than 300 international Journal and conference papers in the areas of databases, mobile computing, peer-to-peer systems, WWW, and wireless networking. He served and is serving as a Program Chair of IEEE International Conferences on Mobile Data Management (MDM'06 and 10) and Advanced Information Networking and Applications (AINA'09 and 14), and IEEE International Symposium on Reliable Distributed Systems (SRDS'12). He guest edited IEEE Journal on Selected Areas in Communications, Sp. Issues on Peer-to-Peer Communications and Applications. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a senior member of IEEE and ACM and a member of three other learned societies.

Shojiro Nishio received his B.E., M.E., and Ph.D. degrees from Kyoto University in Japan, in 1975, 1977, and 1980, respectively. He has been a full professor at Osaka University since August 1992, and was given a peculiar title "Distinguished Professor of Osaka University" in July 2013. He served as a Vice President and Trustee of Osaka University from August 2007 to August 2011. Dr. Nishio has co-authored or co-edited more than 55 books, and authored or co-authored more than 600 refereed journal or conference papers. He served as the Program Committee Co-Chairs for several international conferences including DOOD 1989, VLDB 1995, and IEEE ICDE 2005. He has served and is currently serving as an editor of several international journals including IEEE Trans. on Knowledge and Data Engineering, VLDB Journal, ACM Trans. on Internet Technology, and Data $\&$ Knowledge Engineering. Dr. Nishio has received numerous awards during his research career, including the Medal with Purple Ribbon from the Japanese Emperor in 2011. He is also a fellow of IEEE, IEICE and IPSJ, and is a member of five learned societies, including ACM.