Mobile Information Systems 6 (2010) 5–26
DOI 10.3233/MIS-2010-0090
IOS Press

Global consistency management methods based on escrow approaches in mobile ad hoc networks

Takahiro Hara* and Shojiro Nishio
Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

Abstract. In a mobile ad hoc network, consistency management of data operations on replicas is a crucial issue for system performance. In our previous work, we classified several primitive consistency levels according to the requirements from applications and provided protocols to realize them. In this paper, we assume special types of applications in which the instances of each data item can be partitioned and propose two consistency management protocols which are combinations of an escrow method and our previously proposed protocols. We also report simulation results to investigate the characteristics of these protocols in a mobile ad hoc network. From the simulation results, we confirm that the protocols proposed in this paper drastically improve data availability and reduce the traffic for data operations while maintaining the global consistency in the entire network.

Keywords: Consistency management, mobile ad hoc network, data replication, escrow

1. Introduction

In a *mobile ad hoc network* (*MANET*), two or more mobile hosts can form a temporary network without the need of any existing network infrastructure or centralized administration [2,3,9]. In MANETs, each mobile host acts as a router and communicates with other hosts in a multi-hop manner. MANETs are actively used in military applications, rescue services, and sensor networks but many other applications are expected in the near future.

In MANETs, as mobile hosts move freely, disconnections often occur. This causes data in two separated networks to become inaccessible to each other. Preventing the deterioration of data availability at the point of network partitioning is a very significant issue in MANETs [6,16,17]. To improve data availability in such an environment, data replication is the most promising solution. Based on this idea, we have designed effective data replication techniques in MANETs in our previous papers [17–19]. In [18,19], we have proposed replication methods in an environment where data items are updated. In these methods, we assumed an optimistic consistency management manner in which data operations are tentatively performed on any replicas and the consistency is checked afterward when the operation issued hosts connect to the hosts holding the original data (primary copy). However, such an optimistic

*Corresponding author. Tel.: +81 6 6879 4511; Fax: +81 6 6879 4514; E-mail: hara@ist.osaka-u.ac.jp.

approach may not work well in MANETs due to frequent conflicts of data operations performed in partitioned networks.

Therefore, in [20], we discussed pessimistic consistency management manners for data operations on replicas in MANETs, which determines the success or fail of data operations on the spot. Specifically, we defined several different consistency levels according to application requirements, and then, we proposed protocols to realize them. These consistency levels differ with each other in terms of temporal and spatial aspects. Since consistency management of data operations in MANETs is a very complex issue, we consider our approaches in [20] as the first step of this issue, thus the proposed protocols are very simple. Therefore, various other protocol designs could be considered to further improve the system performance.

For example, in GC (Global Consistency) in which consistency is kept in the entire network, the whole area is divided into several sub-areas called *regions* and proxies in the regions cooperatively behave in order to guarantee that every read operation on data items can read the replicas of the latest version. For this aim, the protocol of GC proposed in [20] uses a *quorum system* [1], which can perform read and write operations even if some mobile hosts that hold replicas disconnect from the network or network partitioning occurs. Although this approach improves data availability in MANETs, it does not work well in an environment where the networks is too sparse as reported in [20]. In such a case, some extensions should be considered to improve data availability. For this aim, a typical approach is weakening the strictness of consistency by adopting a relaxed consistency condition such as *Epsilon serializability* [8] and d*-consistency* [4].

In this paper, we address another effective approach by assuming special types of applications. Specifically, we adopt an *escrow method* [13], which divides the instances of each data item to multiple disjunct partitions and performs data operations locally on each partition. We propose two protocols to achieve GC, which are combinations of an escrow method and protocols proposed in [20]. Here, it should be noted that the protocols proposed in this paper are not very novel in terms of the technical aspect because these are basically simple combinatorial approaches of an escrow method and our previous protocols. The main contributions of this paper are not only the proposal of the protocols, but also (i) providing several different protocol designs which are aware of the characteristics of MANETs and the assumed system model and (ii) extensive performance studies of these protocols. We should note that some of the results of this paper have been reported in [21].

The remainder of this paper is organized as follows. In Section 2, we introduce some related work. In Section 3, we explain the system model. In Section 4, we explain several consistency levels defined in [20]. We present the proposed consistency management protocols based on an escrow method in Section 5. In Section 6, we show the simulation results. Finally we conclude this paper in Section 7.

2. Related work

Consistency management is a popular research topic in distributed database systems. For example, Alonso et al. [14] discussed cache coherency issues in distributed systems and classified several coherency conditions such as time-based, value-based, and version-based ones. The consistency levels which we defined in [20] are basically based on such conventional approaches. There have been also many conventional works that aim to weaken the consistency such as *Epsilon serializability*, which is a generalization of classic serializability and allows some limited amount of inconsistency [8]. As mentioned in Section 1, applying these conventional approaches to weaken the consistency strictness of GC can improve data availability, however, this is not a focus in terms of the scope of this paper.

In mobile environments (not MANET), various consistency management strategies that consider host disconnections have been proposed [4,15,22]. Most of them assume that mobile hosts access databases at sites in a fixed network, and replicate data on the mobile hosts because wireless communication is more expensive than wired. They address the consistency management issue of data operations with low communication costs. These strategies assume only one-hop wireless communication.

In [4], the authors proposed the concept of *d-consistency*, which is a cluster-based approach and allows certain degree of divergence of values of copies in different clusters. Similar to Epsilon serializability [14], this concept is applicable to some of the consistency levels which we proposed in [20], while it is beyond the scope of this paper.

In [10,11], the authors discussed transaction management approaches based on an escrow method in mobile environments. In [10], the authors assume special types of applications in which the instances of each data item can be partitioned among sites, e.g., mobile users share consumable resources. Data operations to these items are basically either references to the number of available instances of an item (read) or consumption of instances (write). A good example is a case that salespersons are selling items from inventory, e.g., medical item supply. In an escrow method (or a site escrow method), the total number of available instances of each item is partitioned across the number of sites (servers) in the system. This can be thought of as each site holding a number of instances in escrow. A transaction issued by a user runs at only one site, and can successfully complete at the site only if the number of instances it requires does not exceed the number of instances available in escrow at the site. In other words, while every transaction can be executed locally at one site independently of other sites, transaction consistency can be maintained globally in the system. The escrow method is very effective in mobile environments where mobile hosts (sites) frequently disconnects from the network. While this method does not assume MANETs, the concept is also applicable to MANETs. In this paper, we adopt an escrow method in combination with consistency management approaches for MANETs.

Recently, data replication is becoming more popular and significant in MANETs [5,12]. Several methods have been proposed for preserving consistency of data operations on replicas in MANETs [6,7]. In [6,7], the authors proposed methods by which replicas are allocated to a fixed number of mobile hosts that act as servers and keep the consistency of data operations. In there, the consistency is maintained by employing a strategy based on a *quorum system* that has been proposed for distributed databases [1]. These methods are considered similar to our GC protocol proposed in [20] because consistency of data operations is maintained based on a quorum system. However, the methods in [6,7] are based on a best-effort approach and cannot strictly keep the consistency in the entire network.

3. System model

In this paper, we assume an environment where each mobile host accesses data items held by other mobile hosts in a MANET and each mobile host allocates replicas of the data items on its memory space. We also assume that the area in which mobile hosts can move around is divided into several regions and the consistency of data operations on replicas is managed based on the regions. Details of the system model are as follows:

– The set of all regions in the entire area is denoted by $R = \{R_1, R_2, \cdots, R_l\}$, where l is the total number of regions and R_r $(r = 1, \cdots, l)$ is the region identifier. We do not restrict to any particular architecture design for regions because in a real situation, regions are geographically defined according to requirements from the application. It may be shared-nothing rectangles or arbitrary circles.

– The set of all mobile hosts in the entire MANET is denoted by $M = \{M_1, M_2, \cdots, M_m\}$, where m is the total number of mobile hosts and M_i $(i = 1, \cdots, m)$ is the host identifier. Each mobile host knows its current location by using some devices such as GPS, and moves around in the given area. There are two kinds of mobile hosts; proxies and peers. A proxy is a specially designated peer who manages other peers in a specific region in the MANET. A proxy has limited movement and does not go out of its region. It also does not encounter with a node failure. (Note that we can easily remove these assumptions by adopting some existing approaches for turning over the role of the proxy by some other peer in the region.) For other peers, there are two kinds of movement according to the application. One is the same as proxy, i.e., limited movement inside the region. The other is unlimited movement, i.e., peers can move across regions. In the latter case, when a peer changes its existing region, it has to register its exit from the current region and its entrance into the new region to the proxies of both regions. In this paper, we basically assume the former case, but also discuss how to deal with the latter case.
  Each proxy knows all other proxies in the entire network. Each peer also knows all the proxies. In a real environment, it can be easily achieved because to join the MANET a new peer has to register its participation to the proxy, and the peer can get the information on all proxies from the proxy. Here, every proxy can know each other at the configuration phase of the MANET.
– Peers communicate with others using wireless communication to construct a MANET. Messages and data items are exchanged between peers using an underlying routing protocol. We do not restrict the routing protocol and any existing protocols can be applied to our assumed system model. Here, proxies may not be within direct communication range of their neighboring proxies, In this case, communication packets are forwarded via other peers in a multihop manner.
– Data is handled as a collection of data items. We assign a unique *data identifier* to each data item located in the system. The set of all data items is denoted by $\boldsymbol{D} = \{D_1, D_2, \cdots, D_n\}$, where n is the total number of data items and D_j $(1 \leqslant j \leqslant n)$ is a data identifier.
– We assume special types of applications in which the mobile users share consumable resources whose instances are indistinguishable in a MANET and the unit of data operation is a pair of read and write operations to check the availability of a necessary number of instances for a real world operation (read) and to reduce/consume the number (write) if they are available. A good example is rescue operations in which consumable items such as medical products and foods are provided to victims based on the decisions of each rescue member. We represent the total number of instances of data item D_j as t_j.
– We assume a simple transaction model in which each transaction consists of a single unit of operation, i.e., a pair of read and write operations. This assumption is based on the fact that many MANET applications require only simple transactions. Of course, there are other applications in which a transaction consists of many data operations. We do not consider such complex transactions in our work here.
  The transaction consistency, i.e., consistency of data operations, is maintained in the entire network, which is defined as GC in [20].
– There are basically three different cases for memory available at proxies and peers for creating replicas. The first case is where proxies and peers have unlimited memory space and they replicate all data items in the entire network. Some conventional works [6,7] also made this assumption. When data items of small sizes such as location and statistical data are shared, this assumption is reasonable.

The second case is where only proxies have unlimited memory space and peers have no memory space. This is a typical case where proxies have much more resources and computational power than other peers.
The last case is where proxies and peers have limited memory space. This is a more general assumption.
Our protocols proposed in this paper can work in all the above cases. However, in the performance evaluation in Section 6, we mainly assume the first case because a data item is the number of available instances of a resource, thus, its data size is small. We also assume the second case to examine the characteristics of our approach.

4. Consistency levels defined in our previous work

In this section, we present some primitive consistency levels for MANETs which we defined in [20] and the protocol to achieve GC.

4.1. Primitive consistency levels: GC, LC, and PC

Since there are many kinds of applications in MANETs, there cannot be one universal optimal strategy for consistency management. Thus, in [20], we proposed four different primitive consistency levels. In this section, we present three consistency levels, GC, LC, and PC, which differ with each other in terms of spatial areas the consistency strictness is required.

4.1.1. Global Consistency (*GC*)

The consistency of data operations on replicas is required in the entire network. Formally, GC requires that every read operation issued by any peer necessarily reads a replica of the latest version in the entire network, i.e., a replica which was written by the latest write operation issued in the entire network. In MANETs, such consistency is hard to achieve, however, there exist some applications that require GC.

A good example is a situation in which members of a rescue team are divided into several groups each of which is responsible of a certain region and the information on the progress of tasks is shared in the entire network. In this case, the shared information is used for administrative decisions at the highest level such as allocation of human resources and scheduling of new tasks. Another example is the case where members of a rescue team share limited resources such as bottles of water and medicines, and the numbers of the resources that are currently available are managed as data items among the members. The members can use or reserve some resources for a rescue operation if the necessary number of resources are available at the time. In the above two cases, the consistency of data operations must be maintained strictly in the entire network.

4.1.2. Local Consistency (*LC*)

The consistency of data operations on replicas is required only in each region. Thus, this consistency level weakens the strictness of consistency from the spatial perspective. Formally, LC requires that in each region, every read operation issued by any peer in the region necessarily reads a replica of the latest version in the region, i.e., a replica which was written by the latest write operation issued in the region. An example of an application that requires LC is a situation in which members of a rescue service team share the information on the damages such as the numbers of injured persons and destroyed buildings, each of which can be separate data items. This information is counted and referenced locally in each region.

4.1.3. Peer-based Consistency (PC)

The consistency of data operations on replicas is required only in each peer. Thus, the strictness of consistency is further weakened. Formally, PC requires that every read operation issued by a peer necessarily reads its own replica to which the latest write operation was performed by itself. An example of an application that requires PC is a situation in which members in a rescue service conduct a survey on some objects, where the survey results (reviews) are recorded as data items. In this case, a member can read and update his own review on each object anytime.

4.2. Protocol to achieve GC

Since we assume GC in this paper, we briefly present the protocol to achieve GC which we proposed in [20].

4.2.1. Basic idea

The simplest way to achieve GC is "read-one write-all", which requires that every write operation is performed on all the replicas. However, it works badly in MANETs due to frequent peer disconnections and network partitioning. Alternatively, we considered a quorum system similar to [6,7].

In a quorum system, read and write operations are performed on all replicas held by mobile hosts that form read and write quorums, respectively, where every pair of read and write quorums have an intersection. Specifically, after a write operation is performed, all mobile hosts in the write quorum hold the latest version of replicas. As for a read operation, since there is an intersection between write and read quorums, at least one mobile host in a read quorum holds a replica of the latest version. Thus, the consistency of data operations on replicas can be kept by reading the latest version.

The quorum based consistency management is suitable for MANETs because it can perform read and write operations when mobile hosts that form quorums are accessible, even if some other mobile hosts that hold replicas disconnect from the network or network partitioning occurs.

4.2.2. Overview

Based on the above idea, we employ a quorum system based on dynamic quorums similar to [7], where mobile hosts are dynamically grouped into quorums, thus, it is tolerant to unpredictable network topology change and node failures in MANETs. The consistency of data operations on replicas is hierarchically managed at two levels; among peers in each region (*local quorum*) and among proxies (*global quorum*). In the following, we explain the details.

First, the quorum size for write operation (to any data item), $|QW|$, and that for read operation, $|QR|$, in the entire network are determined where the condition, $|QW| + |QR| > l$, is satisfied. Here, l is the total number of regions (proxies) in the entire network. Moreover, in each region R_r $(r = 1, \cdots, l)$, the quorum size for write operation on data item D_j, $|QLW_{rj}|$, and that for read operation, $|QLR_{rj}|$, are determined where the condition, $|QLW_{rj}| + |QLR_{rj}| > P_{rj}$, is satisfied. Here, P_{rj} is the total number of peers that hold D_j in the region. If P_{rj} is 1, both $|QLW_{rj}|$ and $|QLR_{rj}|$ are set to 1. Within the conditions, the quorum sizes are arbitrarily determined according to the performance requirements from the application, e.g., $|QR|$ should be set as a small value if read operations are issued much more frequently than write operations.

Owing to condition $|QLW_{rj}| + |QLR_{rj}| > P_{rj}$, every (local) read operation can read the latest version in the region. In addition, owing to condition $|QW| + |QR| > l$, every (global) read operation can read the latest version in the entire network. Thus, a read(write) operation succeeds if it can be performed in $|QR|(|QW|)$ regions, in each of which the operation is performed on $|QLR_{rj}|$ ($|QLW_{rj}|$) replicas.

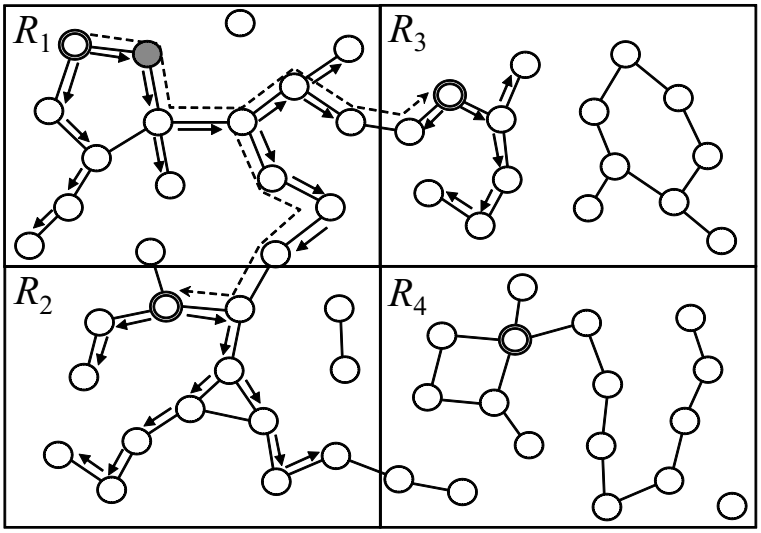


Fig. 1. Message flow in the GC protocol.

4.2.3. Message flow

To cope with network topology change and peer failures during the protocol execution, we adopted a three phase approach for a write(read) operation, which consists of three rounds of message exchanges, (i) a lock request/its reply, (ii) write operations/its acknowledgment (a request for the latest replica/data transfer), and (iii) a message for commit and lock release/its acknowledgment (acknowledgment and lock release). The details of the procedure are shown in [20].

Figure 1 shows an example of message flow in phase (i) of this protocol. In this example, there are four shared-nothing square regions ($R_1, \cdots, R_4$). A circle denotes a peer, whereas a double-lined one denotes a proxy. A solid line between two peers denotes a wireless link. Here, let us suppose that every peer has unlimited memory space and creates replicas of all data items. Thus, P_{rj} is equal to the total number of peers in R_r for all data items, i.e., $\{P_{1,j}, P_{2,j}, P_{3,j}, P_{4,j}\} = \{17, 14, 15, 17\}$. Let us also suppose that $|QR| = 2$, $|QW| = 3$, $|QLR_{rj}| = \lfloor P_{rj}/2 \rfloor$, and $|QLW_{rj}| = P_{rj} - |QLR_{rj}| + 1$.

Here, we assume that the gray colored peer in R_1 issues a request for write operation on data item D_1. In Fig. 1, a solid-lined arrow denotes a message transmission to construct a local quorum in each region and a broken-lined arrow denotes a message transmission to construct a global quorum. In this case, the proxy in R_1 tries to construct a global quorum (the size is 3) for write operation but it fails. This is because it connects to other two proxies in R_2 and R_3, however, the proxy in R_3 fails to construct a local quorum in its region. As shown in this figure, this GC protocol requires a large number of message transmissions and does not work well if the network density is not very high.

5. GC protocols based on an escrow method

In this section, we present two GC protocols proposed in this paper. These protocols are combinatorial approaches of an escrow method and protocols proposed in [20].

As reported in [20], the success ratios of read and write operations in the original GC protocol suddenly degrades from a certain point as the density of peers in the network becomes lower. Specifically, in the simulation reported in [20], as the size of the entire area changes from 400[m] $\times$ 600[m] to 600[m] $\times$

900[m] in which 240 peers exist, the success ratios of data operations in GC degrades from almost 100% to 0. This result shows that keeping the global consistency in the entire network is very difficult.

Therefore, to further improve the success ratios of data operations, we adopt a site escrow method presented in [13] to achieve GC in MANETs. In the following, we present the two GC protocols, which differ in how instances of a data item are partitioned into multiple peers or proxies.

5.1. The region escrow (RE) method

The first one is the *Region Escrow* (*RE*) method, in which the total number of instances, t_j, of each data item D_j is partitioned into regions in the MANET. Specifically, each region R_r $(r = 1, \cdots, l)$ escrows j_r instances of data item D_j on condition that $t_j = \sum_{r=1}^{l} j_r$. This partitioning can be performed either at the configuration phase of the MANET or online by applying the redistribution strategy described later. It can be considered that j_r instances are reserved for peers in region R_r. In this paper, we do not restrict to any particular strategy to determine the numbers of instances escrowed to regions, because it is beyond the scope of this paper. We can simply adopt the uniform distribution where $j_r = t_j/l$ or a strategy which distributes instances according to the resource consumption rate in each region.

After partitioning the instances into regions, data operations can be executed in each region independently of other regions. It means that the RE method is almost identical to the LC protocol proposed in [20] while achieving GC. Therefore, it is expected that success ratio of data operations much improves even if the MANET is sparse, and the traffic for data operations is also much reduced. Actually, a simulation result reported in [20] shows that the LC protocol keeps very high success ratio of data operations (more than 80%) even if the network becomes sparse and the success ratio of the GC protocol becomes below 5%.

5.1.1. Data operation procedure

In this clause, we explain the procedure to perform a data operation in the RE method. Here, for easy understanding, we assume that peers do not move beyond their regions.

Basic idea:

In the RE method, for each data item D_j, the number of instances that are currently available in each region R_r is maintained as the value of D_j, and replicated by peers in the region. We represent it as j'_r. Here, j'_r is equal to or less than j_r if redistribution of the instances is not performed.

When a peer in region R_r issues a request to consume x instances of data item D_j as a data operation, the request is processed by using a protocol similar to the LC protocol in [20]. Specifically, the request is processed using a dynamic quorum system inside the region. Here, since we assume that a data operation consists of a pair of read and write operations, we use only one kind of the quorum size for a data operation, $|QL_{rj}|$, which is different from the original LC protocol that uses two kinds of quorum sizes, $|QLR_{rj}|$ and $|QLW_{rj}|$, for read and write operations, respectively.

The quorum size $|QL_{rj}|$ is determined where the condition, $2 \times |QL_{rj}| > P_{rj}$, is satisfied. Here, P_{rj} is the total number of peers that hold a replica of D_j in the region. The simplest way to achieve this is setting $|QL_{rj}|$ as $\lfloor P_{rj}/2 \rfloor + 1$. Owing to condition $2 \times |QL_{rj}| > P_{rj}$, it is guaranteed that every data operation can read the latest version of D_j, i.e., j'_r. Then, if $j'_r \geqslant x$ (an enough number of instances are available for the data operation), the data operation successfully completes by updating j'_r to $j'_r - x$.

Protocol:

We adopt a three phase approach for a data operation, which consists of three rounds of message exchanges, (i) a lock request/its reply with the latest value, (ii) update/its acknowledgment, and (iii) a message for commit and lock release/its acknowledgment.

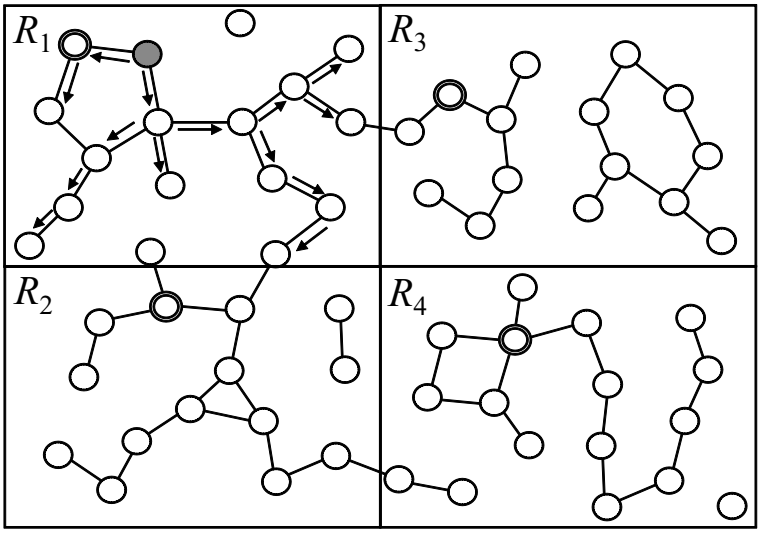


Fig. 2. Message flow in the RE method.

In the following, we describe the details of each of the three phases. For easy understanding of our protocol, we explain them by assuming the limited movement case, where peers do not move beyond its existing region.

– *Phase* (*i*)

When a data operation to consume x instances of data item D_j is issued by a peer in region R_r, the peer tries to set *local locks* to arbitrary $|QL_{rj}|$ replicas of data item D_j held by peers in R_r. If $|QL_{rj}|$ is 1 and the peer holds D_j, it sets the local lock to its holding replicas. Otherwise, if the proxy is the only one that holds a replica, i.e., the second case for memory limitation, the peer unicasts a local lock request to the proxy. Otherwise, the peer multicasts a local lock request to all peers that hold D_j in its region.

Each peer that received the request sets the local lock to its holding replica of D_j and sends back a reply to the request issued peer. The reply contains the information on the version (time stamp) and the value of the replica.

If the peer succeeds to set the necessary number of local locks, i.e., equal to or more than $|QL_{rj}|$ peers or the proxy replied the local lock request, phase (i) succeeds, and then, the procedure goes to the second phase (ii). Otherwise, the operation request fails immediately.

Figure 2 shows an example of message flow in phase (i) of the RE method. In this example, we assume the same situation as the GC protocol shown in Fig. 1. Since a data operation is processed in each region in the RE method, the number of messages generated is much lower than that in the original GC protocol. Moreover, the operation succeeds because the request issued peer can construct a local quorum in its region, while it fails in the original GC protocol.

– *Phase* (*ii*)

The request issued peer checks the latest version (j'_r) of the replica among the received ones. If $j'_r - x < 0$, i.e., x instances are not available in the region, there are two options; the request aborts or the procedure goes to the redistribution process described later.

Otherwise, if $j'_r - x \geqslant 0$, the peer sends the new value of D_j, $j'_r - x$, to $|QL_{rj}|$ peers among those responded in phase (i). Here, if the number of peers with the local lock is more than $|QL_{rj}|$, the peer chooses $|QL_{rj}|$ peers which are closest from the peer. Each peer that received the new value

replaces the old value with the received one, i.e., the write operation is performed, and sends back the acknowledgment to the request issued peer.

Here, it can happen that phase (ii) fails due to the disconnections or failures of peers, i.e., less than $|QL_{rj}|$ peers having locked replicas are accessible from the peer. In such a case, the request aborts.

– *Phase* (*iii*)

If the request issued peer receives the acknowledgment from all the $|QL_{rj}|$ peers, it sends a commit and lock release message to the peers that replied the local lock request in phase (i). Then, each peer that received the message sends back the acknowledgment to the request issued peer and releases the local lock set to its holding replica.

If the request issued peer receives the acknowledgment from all the peers having locked replicas, the operation successfully completes. If phase (iii) fails during the execution, the operation fails and the request issued peer sends an abort message to all the peers having locked replicas.

– *Timeout in processing*

Each of the three phases (i), (ii), and (iii) has the timeout at both request issued peer and other peers. As for the request issued peer timeout, if it cannot receive the required number of replies within the timeout, it aborts the operation. As for the other peers timeout, if a peer has not received a message for the next procedure after it replied to the request issued peer for the previous phase, it aborts the entire procedures performed for the corresponding operation.

5.1.2. Redistribution of instances

In an escrow method, redistribution of instances often occurs according to several requirements. For example, if some new resources become available at the application level, e.g., some medicines are newly supplied for victims, the corresponding data item should be updated by redistributing its instances. Also, if peers in a region run out of the instances of a certain data item, redistribution of instances should be performed with another region.

In this clause, we explain the redistribution process in the RE method. While there are many choices for this aim, we choose a simple one. Here, let us denote the proxy which issues the redistribution process as *the requester* and denote the proxy in the region which is requested by the requester to change the number of instances as *the recipient*. In a case when new resources become available, the requester is the proxy of the region at which the new resources arrive, and the recipient is the proxy of the region in which some of the new resources (suppose y resources) are distributed. In a case when peers in a region run out of the instances, i.e., $j'_r < x$, the requester is the proxy of the region and the recipient is the proxy of the region in which an enough number of instances are available and some of them can be moved to the region of the requester.

In the former case (distributing new resources), the redistribution process is identical to the case that the requester requests the recipient to execute a data operation that consumes $-y$ resources in the recipient's region.

In the latter case (running out of the instances), the redistribution process is as follows. From the closest proxy to farther one, the requester in region R_q successively sends a redistribution request to the proxy until it finds the proxy of the region (R_c) that meets the condition $(j'_q + j'_c)/2 \geqslant x$. This condition prevents taking away instances from a region that has not many instances. Specifically, each proxy (in R_r) that receives the request executes phase (i) of a data operation and checks whether $(j'_q + j'_r)/2 \geqslant x$. If $(j'_q + j'_r)/2 \geqslant x$, the proxy becomes the recipient of the redistribution process, i.e., $R_r = R_c$. Then, the proxy restarts the data operation from phase (ii), where the number of consumed resources is set as $j'_c - (j'_q + j'_c)/2$. At the same time, the requester also executes a data operation where the number

of consumed resources is set as $x - (j'_c - (j'_q + j'_c)/2)$. Here, this operation can start from phase (ii) because the requester already executed phase (i) in the original data operation process. After completing the above procedures, the corresponding data operation successfully completes.

If all these procedures fail, the redistribution process fails and the corresponding data operation aborts.

Here, it should be noted that because the redistribution process is performed between proxies, each proxy can know the current status of instances in its region in most cases. More specifically, it can recognize if its region runs out of instances. Therefore, the proxies in the regions which run out of instances can immediately refuse the redistribution request from the requester without broadcasting the request for phase (i) in its region, which avoids generating useless messages.

5.2. The Peer Escrow (PE) method

The second protocol is the *Peer Escrow* (*RE*) method, in which the total number of instances, t_j, of each data item D_j is partitioned into peers in the MANET. Specifically, each peer M_i $(i = 1, \cdots, m)$ escrows j_i instances of data item D_j on condition that $t_j = \sum_{i=1}^{m} j_i$. This partitioning can be performed either at the configuration phase of the MANET or online by applying the redistribution strategy. Here, similar to the RE method, we do not restrict to any particular strategy to determine the numbers of instances escrowed to peers.

5.2.1. Data operation procedure

In this clause, we explain the procedure to perform a data operation in the PE method. For each data item D_j, the number of instances that are currently available at each peer M_i is maintained as the value of D_j. We represent it as j'_i. Here, j'_i is equal to or less than j_i if redistribution of the instances is not performed.

When a peer M_i issues a request to consume x instances of data item D_j as a data operation, the request is processed locally at the peer (without any message transmissions with other peers) similar to the PC protocol in [20]. Specifically, the request issued peer checks the value of D_j (j'_i). If $j'_i - x < 0$, the data operation fails or the procedure goes to the redistribution process. Otherwise, if $j'_i - x \geqslant 0$, the data operation successfully completes by updating j'_i to $j'_i - x$.

5.2.2. Redistribution of instances

In this clause, we explain the redistribution process in the PE method. In the PE method, the redistribution process also occurs when distributing new resources and running out of the instances. However, in both cases, the requester and the recipient are not proxies but general peers. In the case of distributing new (y) resources, the redistribution process is identical to the case that the value of the corresponding data item held by the recipient is increased by y.

In the latter case of running out of the instances, the redistribution process is as follows. First, the requester M_q (request issued peer of the corresponding data operation) that runs out of the instances broadcasts a redistribution request in its existing region. Here, the request is attached with the information on x, the number of instances consumed in the operation. Each peer M_i that receives the request sends back a reply attached with the information on j'_i if it meets the condition $(j'_q + j'_i)/2 \geqslant x$. Among the peers that sent back a reply, the requester chooses the recipient M_c whose j'_i is the largest among them. Then, j'_q is replaced by $(j'_q + j'_c)/2 - x$ at M_q and j'_c is replaced by $(j'_q + j'_c)/2$ at M_c. After that, the corresponding data operation successfully completes.

If the requester does not receive any replies, from the closest proxy to farther one, it successively sends a redistribution request to the proxy, and the same procedure described above is performed in the

corresponding region. This repeats until the requester receives a reply from a peer. If it also does not receive any replies, the redistribution process fails and the data operation aborts.

It should be noted that each peer blindly performs the above redistribution process, while in the RE method, proxies performs the redistribution process in a systematic manner. Therefore, this process may cause large amount of traffic if the number of remaining instances becomes very low in the entire network.

5.3. How to deal with peers' movement across regions

The PE method works in an environment where peers move across regions without modification, because it basically runs locally at each peer. Even in the redistribution process, it only needs to recognize its existing region.

On the other hand, the RE method uses the number of peers in each region, P_{rj}, to construct local quorums, which dynamically changes when peers move across regions. Because the proxy manages the current value of P_{rj}, the RE method must be modified so that every data operation is processed via the proxy. Let us recall that a data operation consists of a pair of read and write operations. Here, for easy understanding, we separately consider the local quorum size for read and write operations as $|QLR_{rj}|$ and $|QLW_{rj}|$, while in the RE method $|QLR_{rj}| = |QLW_{rj}|(= |QL_{rj}|)$.

As for a write operation, the local quorum size ($|QLW_{rj}|$) in the region can be dynamically determined based on the current value of P_{rj}. On the other hand, a read operation must consider the local quorum to which the latest write operation was performed in the region, whereas the members might have changed a lot.

A possible way to solve this problem is recording at the proxy the time T'_j when the latest write operation to D_j was performed in the region, and the number of peers P'_{rj} that hold D_j in the region at T'_j. By using P'_{rj}, the proxy can calculates $|QLW'_{rj}|$, which is the local quorum size for the latest write operation. Then, the quorum size for read operation in the region is initially set as $|QLR_{rj}| = P'_{rj} - |QLW'_{rj}| + 1$ to guarantee that every pair of local write and read quorums have an intersection. After then, $|QLR_{rj}|$ is decreased by one every time when a peer having a replica of D_j, which has belonged to the region before and after the latest write operation was issued but not being involved in the local write quorum, exits from the region. This can be done by using the information recorded at the proxy on the latest write time T'_j, that on the time tp when the peer came into the region, and that on the version of the replica of D_j held by the peer. By doing so, every local read quorum includes at least one peer that holds a replica to which the latest write operation was performed.

When a data operation, a pair of read and write operation, to D_j is issued by a peer, the operation succeeds if the proxy successfully receives replies with data values in phase (i) from more than $|QLR_{rj}|$ peers that have belonged to the region before and after the latest write time, i.e., $T'_j > tp$, and it successfully performs a write operation on replicas of more than $|QLW_{rj}|(= |QL_{rj}|)$ peers in the region.

6. Simulation

In this section, we show results of simulation experiments to evaluate our proposed RE and PE methods. For comparison, we also show the performance of the GC protocol proposed in [20], which we call the GC method. Since the limitation of memory space may affect the system performance, we evaluated the

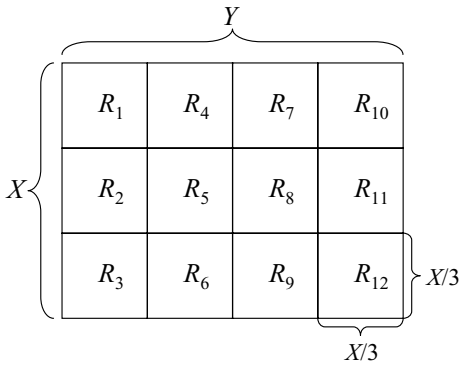


Fig. 3. Simulation area.

two cases discussed in Section 3; *Case 1:* unlimited memory space for both proxies and peers and *Case 2:* unlimited memory space for proxies.

For simplicity, we examined only the performance of the case of the limited mobility inside region for peers. We also assumed that neither network topology change nor peer failure occurs during executing the protocol in every method. These assumptions are not always true in a real situation, but we confirmed from some preliminary experiments that the features of the evaluation results were basically same in the cases with and without these assumptions, and the differences between them are not significant. Moreover, these assumptions are advantageous to the GC method rather than the RE and PE methods because the GC method requires message exchanges in the entire network and its performance drops when a network is highly dynamic.

6.1. Simulation model

In our simulation experiments, we assume a situation where members engaged in a collaborative work such as rescue operations share consumable resources. The members are divided into groups each of which is assigned a specific region. They are equipped with mobile terminals with a wireless communication facility such as Bluetooth, wireless LAN, and Zigbee. In the experiments, we assume that a unit of simulation time corresponds to 5 [s] in a real environment. In the following, we present the details of the simulation model.

Mobile hosts exist in an area of $X \times Y$ [m^2] which consists of 12 regions of $X/3 \times X/3$, $R = \{R_1, \cdots, R_{12}\}$ (See Fig. 3). Here, ratio $X : Y$ is kept to $3 : 4$, i.e., Y is automatically determined if X is determined. In our experiments, X is changed as a variable parameter in the range from 300 [m] to 600 [m]. Here, the experiments changing X are almost identical to those varying the number of mobile hosts and the radio communication range because all of them affect the connectivity among mobile hosts.

The number of mobile hosts in the entire system is 240 ($M_1, M_2, \cdots, M_{240}$). M_i ($i = 1, \cdots, 12$) is the proxy of region R_i, and M_j ($j = 13, \cdots, 240$) is a peer that exists in region $R_{(m \bmod 12)}$ if (m mod 12) is not 0 or R_{12} if (m mod 12) is 0. That is, each region contains static 20 mobile hosts. Each peer moves according to the random waypoint model, where each host selects a random destination in its assigned region. The pause time and the maximum movement speed are set as 0 [s] and 2 [m/s], respectively, assuming that peers move in a walking speed.

The communication range of each mobile host is a circle with radius 50 [m]. To remove the impact of underlying network protocols, we simply assume that every message and data transmission is routed

via the shortest path from the source to the destination. Moreover, even in a multicast transmission, e.g., lock request, a message or data item is separately transmitted to each of the multicast members via the shortest path, i.e., a multicast transmission consists of separate unicast transmissions.

Since we assume unlimited memory space for peers or proxies, the number of data items in the entire network is not a parameter that affects the performance. Therefore, we just focus on a data item, D_1. The frequency of data operations issued at each proxy and peer is basically set as 0.002 [1/s]. That is, every proxy and peer has the same operation issue frequency. In an experiment, we also assume another case in which the frequency is set as 0.0038 for peers in regions $R_1, R_3, \cdots, R_{11}$, and as 0.0002 for peers in regions $R_2, R_4, \cdots, R_{12}$. This represents a situation in which there are two kinds of regions, hot and cold, i.e., operation issue frequencies are skewed.

In the GC method, the size of the global quorum, $|Q|$ ($= |QW| = |QR|$), is set to 7 ($= 12/2 + 1$). As mentioned in Section 3, we assume two cases for memory limitation; *no memory limitation for all* in which all peers replicate the data item and *no memory limitation for proxies* in which all proxies replicate the data item but peers do not. Thus, in the GC and RE methods, the size of the local quorum, $|QL_{rj}|$ $(r = 1, \cdots, 12,\ j = 1)$, is set as $\lfloor P_{rj}/2 \rfloor + 1 = 11$ ($= 20/2 + 1$) in the first case and as 1 in the second case.

In the simulations, we measured the *success ratio* and the *traffic* during the simulation time. The success ratio is defined as the ratio of successful data operations to the number of all requests of data operations issued during the simulation time. The traffic is defined as the average of the total hop count for message exchanges to process a data operation including transmissions of data items (values of instances), i.e., average hopcount for message transmission per request. Here, for simplicity we assume that all the messages and data items are of the same size.

6.2. Impact of area size – steady state

We first examine the impact of the area size on the performance of the three methods in steady states. Specifically, we measured the success ratio and the traffic in two different cases where an enough number of instances are available (the initial number of instances in the entire network is set as 2.4×10^8) and no instance is available (the initial number is set as 0). In the first case, we assumed uniform distribution of instances in the RE and PE methods, where each region escrows 2×10^7 instances in the RE method and each peer escrows 1×10^6 instances in the PE method. The simulation in the first case aimed to verify the effectiveness of our escrow approaches in improving data availability and reducing message traffic. It should be noted that in this case a data operation request fails only when the network connectivity is low and the request issue peer cannot set the necessary number of locks to construct a quorum in the GC and RE methods. It also should be noted that redistribution of instances does not occur in both of the RE and PE methods. The simulation in the second case aimed to examine the behavior of the system when it runs out of instances and also to examine the impact of redistribution of instances on the traffic.

The simulations were run during 200,000 units of simulation time (1,000,000 [s]).

6.2.1. The case of having enough instances

Figure 4 shows the simulation result in the case that enough instances are available. In both graphs, the horizontal axis indicates the area size, X. The vertical axis indicates the success ratio in the case of (a) and the traffic in the case of (b). In these graphs, '(p)' represents the case of *no memory limitation for proxies*, and the others represent the cases of *no memory limitation for all*. Note that the PE method cannot be applied in the case of no memory limitation for proxies.

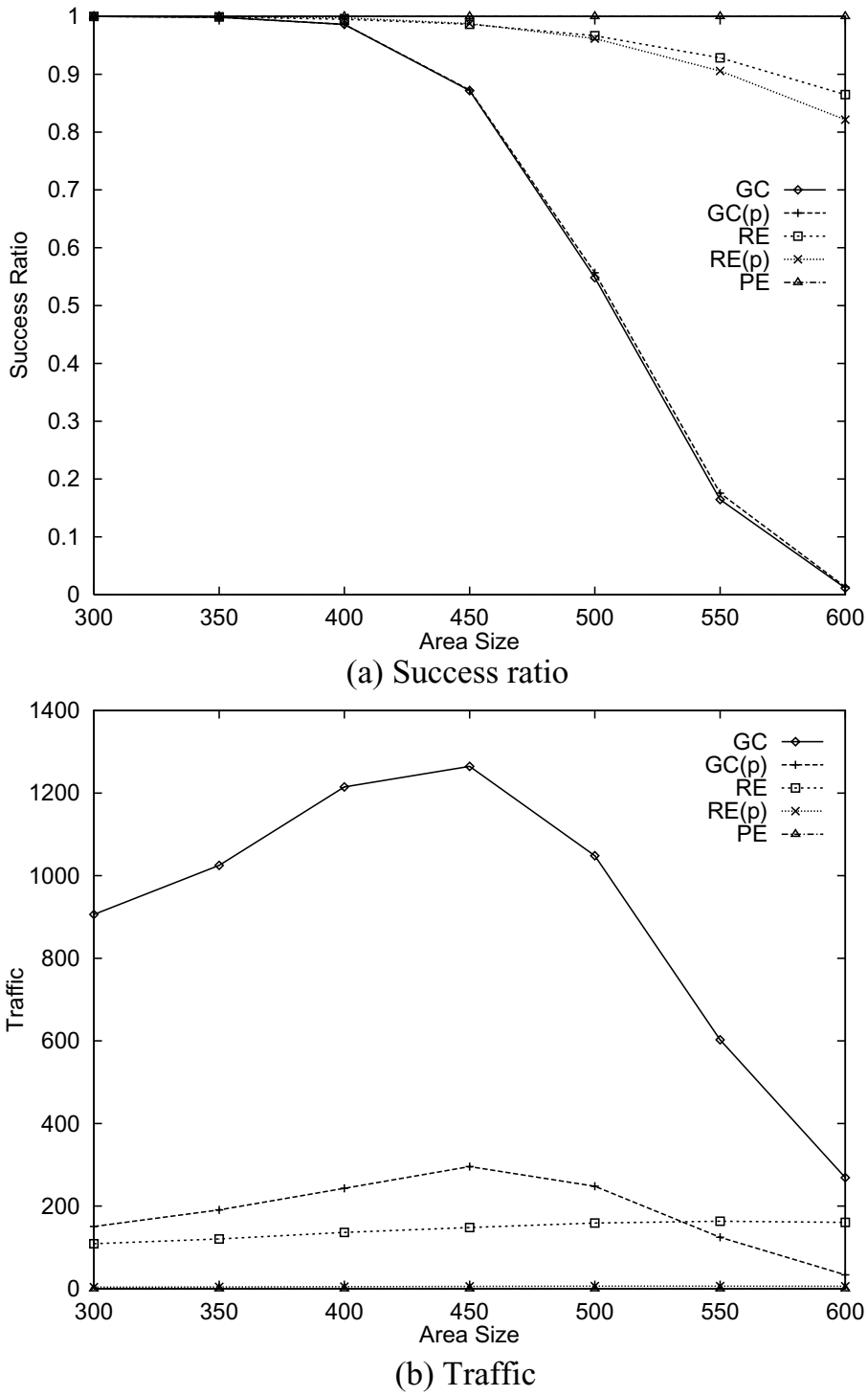


Fig. 4. Effect of area size (The number of instances is enough).

From Fig. 4(a), the success ratios in GC and RE get lower as the area size gets larger. This is because the connectivity among mobile hosts becomes lower, and thus, the proxy cannot set the necessary number of locks on replicas with high probability. We can see a remarkable feature that when the area size is larger than 450, the success ratio in GC suddenly gets lower in both memory limitation cases but it remains high in RE. This fact shows that even when the connectivity among mobile hosts is still high in each region, the connectivity among proxies becomes low. Thus, we can confirm that an escrow approach to distribute instances to regions is very effective to improve data availability. Moreover, a peer escrow approach, PE, is much more effective because it always achieves 100% of the success ratio when an enough number of instances are available. Comparing the two memory limitation cases for RE, RE in the case of no memory limitation for proxies achieves lower success ratio than that in the other case. This is because in this case data operations sometimes fail due to the disconnection with the proxy even if the request issue peer connects to a large number of peers, i.e., more than $|QL_{rj}| = 11$.

From Fig. 4(b), the traffic produced by GC in both memory limitation cases gets higher and then gets lower from a certain point ($X = 450$) as the area size gets larger. The reason why the traffic first gets higher in GC is that the proxy that received a request of a data operation from a peer often fails to find proxies that can set the necessary number of local locks in their responsible regions. The traffic in GC then gets lower because the connectivity among mobile hosts gets lower, and thus, the number of proxies and peers that a request issued peer can communicate with also becomes lower. This fact can be confirmed from the results in Fig. 4(a) in which the success ratio in GC gets lower. Of the three methods, GC produces the highest traffic, and RE produces much lower than GC. Of course, PE produces no traffic. This result makes us confirm that an escrow approach is very effective to not only improve data availability but also reduce traffic for data operations. It can be also seen that in GC and RE, the case of no memory limitation for proxies produces much lower traffic than the other memory limitation case. This shows that limiting the number of peers that hold replicas is very effective to reduce traffic.

6.2.2. The case of having no instance

Figure 5 shows the simulation result in the case that no instance is available. In this graph, the horizontal and vertical axes indicate the area size, X, and the traffic, respectively. We omit the result of the success ratio because it is obviously zero in every method. In the graph, we separately show the traffic of the RE and PE methods with redistribution of instances denoted by 'RE_re' and 'PE_re' and those without redistribution denoted by 'RE' and 'PE'. '(p)' represents the case of *no memory limitation for proxies*, and the others represent the cases of *no memory limitation for all*.

From this result, we can observe an interesting fact that PE with redistribution produces much more traffic than RE with redistribution. This is because in the RE method, the redistribution process is controlled by proxies and thus the proxies can refuse to act as the recipients without broadcasting a local lock request in their responsible regions if they already run out of instances, which helps reducing the traffic. On the contrary, in the redistribution process in the PE method, since each peer blindly searches a recipient, the request message is broadcast in the entire network even if all peers run out of instances. The same feature as RE with redistribution can be seen in GC. While the redistribution process does not occur in the GC protocol, all data operations are controlled by proxies and thus the proxies that already run out of instances can refuse a global lock request without broadcasting a local lock request in their regions. We can confirm this fact by comparing the traffics produced by GC in Figs 5 and 4(b). In RE, and RE(p) without redistribution, the traffics caused by them are smaller than those in Fig. 4(b), because phase (iii) does not occur due to the lack of instances.

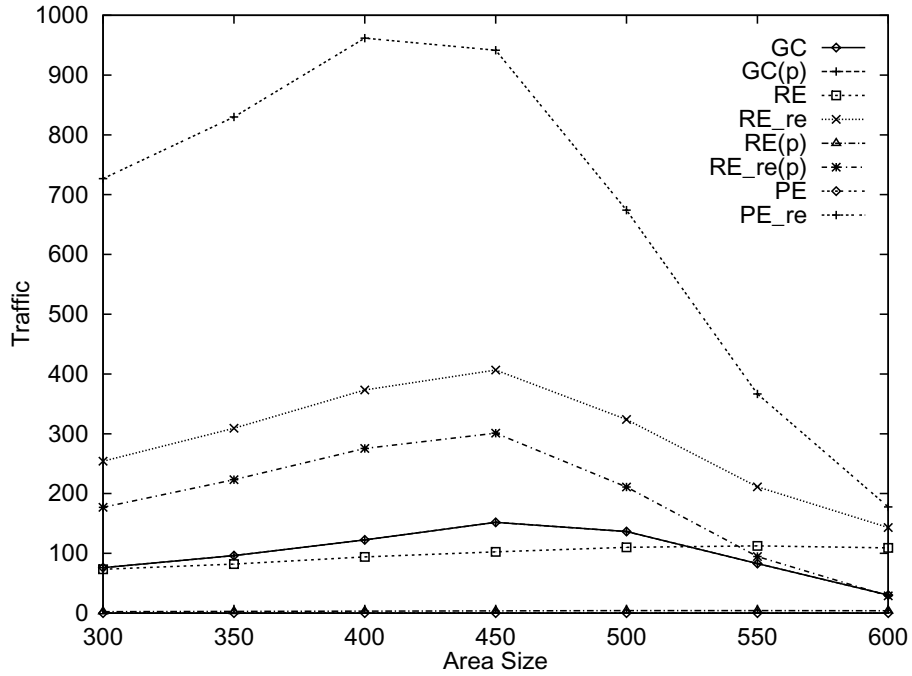


Fig. 5. Area size and traffic (the number of instances is zero).

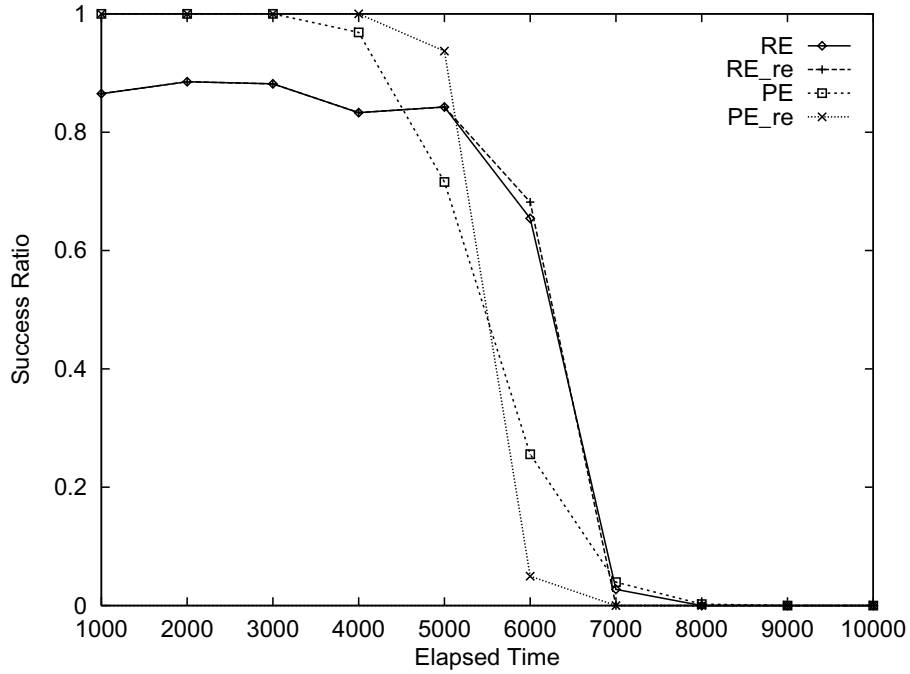
6.3. Performance in a transient state

Next, we examine the behaviors of the three methods in more detail by measuring their performance where the remaining number of instances decreases as time passes. Here, it should be noted that in this simulation we measured the two criteria in a transient state, i.e, the simulation results are not converged. Thus, the values themselves in all the graphs of the simulation results are not very significant but are just for reference to observe the behavior of each method. Specifically, we measured the success ratio and the traffic during 10,000 units of simulation time (50,000 [s]) where the initial number of instances in the entire network is set as 12,000. We assumed uniform distribution of instances in the RE and PE methods, where each region escrows 1,000 instances in the RE method and each peer escrows 50 instances in the PE method. Note that in this experiment, to fairly examine the impact of the redistribution process and compare between the RE and PE methods, we omit the results of the GC method and the RE method in the case of no memory limitation for proxies.

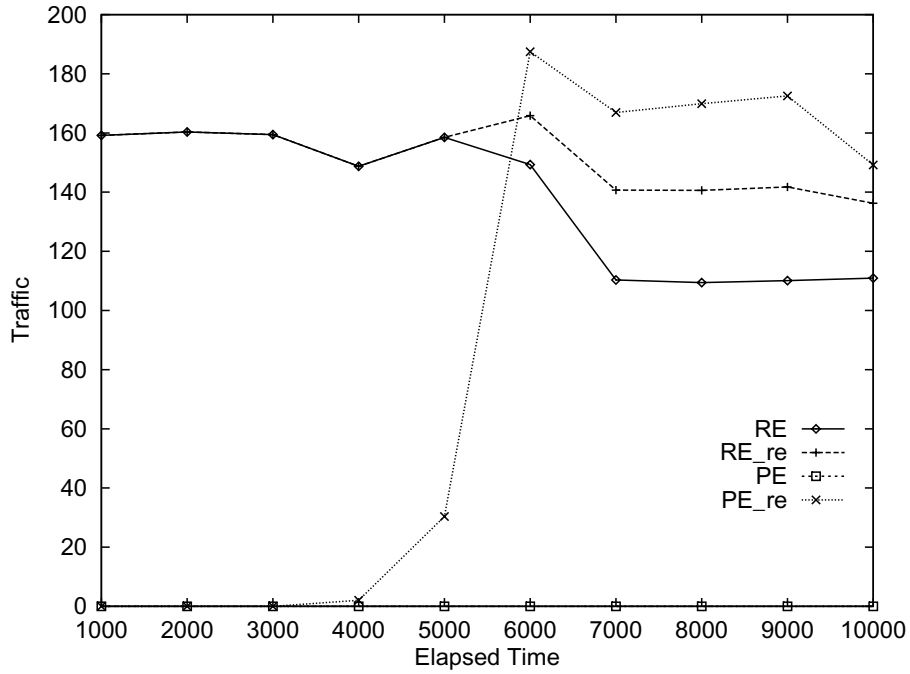
We assumed two different cases for data operation frequencies as described in Section 6.1, (i) the frequency at every proxy and peer is set as 0.002 [1/s] and (ii) the frequency is set as 0.0038 for peers in regions $R_1, R_3, \cdots, R_{11}$, and as 0.0002 for peers in regions $R_2, R_4, \cdots, R_{12}$, i.e., operations are skewed. In the first case, all peers run out of instances at almost the same time, while in the second case, peers in six *hot* regions $R_1, R_3, \cdots, R_{11}$ run out of instances much faster than those in the other regions.

6.3.1. Uniform operation frequency

Figure 6 shows the simulation result in case (i) uniform operation frequency. In both graphs, the horizontal axis indicates the elapsed units of simulation time (not in seconds). The vertical axis indicates the success ratio in the case of (a) and the traffic in the case of (b). Here, these criteria were calculated as the averages during every 1,000 units of simulation time.

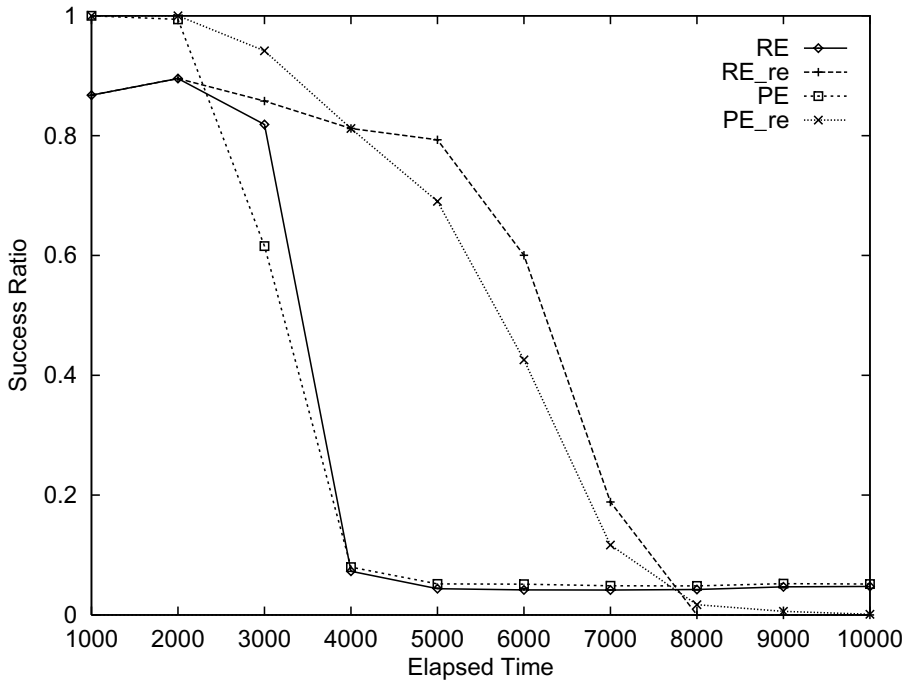


(a) Success ratio

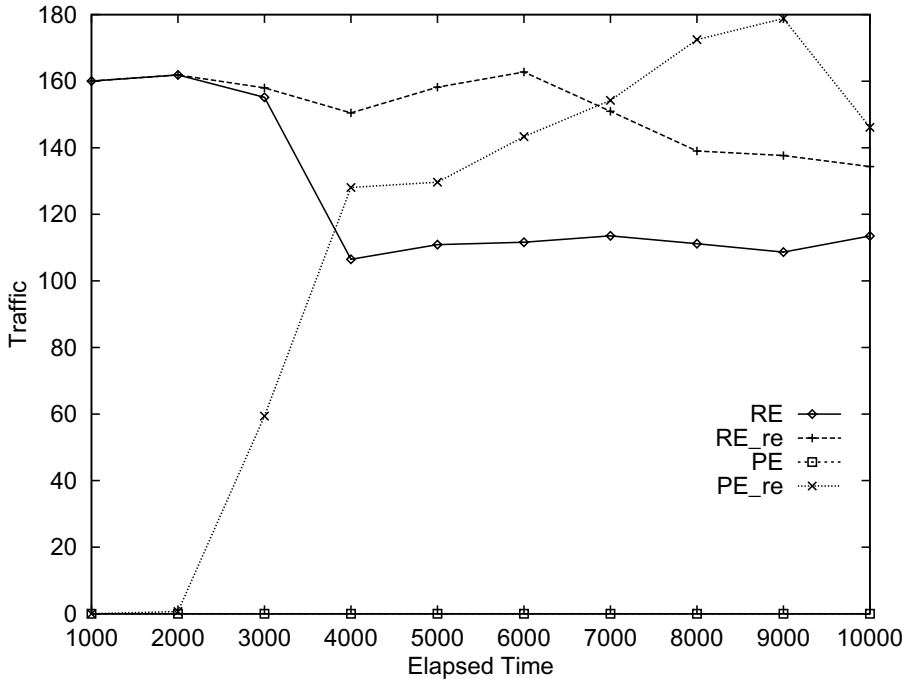


(b) Traffic

Fig. 6. Elapsed time and performance (Uniform access).



(a) Success ratio



(b) Traffic

Fig. 7. Elapsed time and performance (Skewed access).

From Fig. 6(a), in all of the four cases, the success ratio suddenly drops from a certain point as time passes. This is because all peers issue data operation requests with the same frequency, thus, they run out of instances at almost the same time. It is also shown that in the case of uniform operation frequency, the impact of redistribution of instances is not significant in both of the RE and PE methods. In particular, the success ratios in the PE method with and without redistribution are almost same, while the RE method with redistribution makes the performance drop point slightly later. This little impact of redistribution is also due to that they run out of instances at almost the same time.

From Fig. 6(b), it is shown that redistribution of instances starts to occur from a certain point and this increases the traffic. As mentioned in Section 6.2.1, the increase of traffic caused by redistribution in the PE method is much larger than that in the RE method. Here, the traffic produced by the RE method keeps almost a constant for a while as time passes, it then starts decreasing, and lastly it again keeps almost a constant. This is due to the difference in traffic caused by a successful operation (three rounds of message exchanges) and a failed operation (two rounds).

6.3.2. Skewed operation frequencies

Figure 7 shows the simulation result in case (ii), skewed operation frequencies. In both graphs, the horizontal axis indicates the elapsed simulation time. The vertical axis indicates the success ratio in the case of (a) and the traffic in the case of (b).

The result shows similar features to that in Fig. 6. However, the effectiveness of redistribution of instances is much higher than that in Fig. 6. Specifically, redistribution makes the drop points of success ratio much later in both of the RE and PE methods. This is due to the big difference in operation issuing frequencies among peers or regions, i.e., peers or regions that run out of instances can take over instances from those with low operation frequency. Here, the traffic produced by the RE method without redistribution keeps almost a constant after the performance drop, because only peers in six regions with low operation frequency succeed the operations and they do not run out of instances during the simulation time.

7. Conclusions

In this paper, we assumed special types of applications in which instances of data items can be partitioned and proposed two protocols, RE and PE, to achieve GC (Global Consistency) in the entire network. The proposed methods are combinations of an escrow method and protocols proposed in [20]. In the RE method, each region escrows the partitioned instances, while in the PE method, each peer escrows the instances.

We also conducted the simulation experiments to verify the effectiveness of our proposed methods. The results showed that an escrow approach is very effective in MANETs for both improving data availability and reducing traffic for data operations. In particular, we confirmed that the PE method works very well when an enough number of instances of a data item are available. We also confirmed that redistribution of instances is very effective for peers that run out of their instances. We found that when most peers run out of their instances, the PE method produces much more traffic than the RE method. This is because in the redistribution process in the PE method, each peer blindly searches a recipient of instances by broadcasting the request message.

As part of our future work, we plan to combine our approach with other existing approaches that weaken the transaction consistency such as *Epsilon serializability* [8] and *d-consistency* [4].

Acknowledgments

This research was partially supported by Grant-in-Aid for for Scientific Research on Priority Areas (18049050) and for Scientific Research (S)(21220002) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, by Mobile Wireless Foundation of Mobile Radio Center, and by the Ministry of Internal Affairs and Communications of Japan under the Research Development Program of "Ubiquitous Service Platform."

References

[1] D. Barbara and H. Garcia-Molina, The reliability of vote mechanisms, *IEEE Transactions on Computers* **36**(10) (1987), 1197–1208.
[2] D.B. Johnson, Routing in ad hoc networks of mobile hosts, In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, 1994, pp. 158–163.
[3] D.J. Baker, J. Wieselthier and A. Ephremides, *A distributed Algorithm for Scheduling the Activation of Links in a Self-Organizing*, mobile, radio network, In Proceedings of IEEE ICC'82, 1982, pp. 2F6.1–2F6.5.
[4] E. Pitoura and B. Bhargava, Data consistency in intermittently connected distributed systems, *IEEE Transactions on Knowledge and Data Eng* **11**(6) (1999), 896–915.
[5] G. Cao, L. Yin and C.R. Das, Cooperative cache-based data access in ad hoc networks, *IEEE Computer* **37**(2) (2004), 32–39.
[6] G. Karumanchi, S. Muralidharan and R. Prakash, *Information Dissemination in Partitionable Mobile Ad Hoc Networks*, In Proceedings of SRDS'99, 1999, pp. 4–13.
[7] J. Luo, J.P. Hubaux and P. Eugster, *PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems*, In Proceedings of ACM MobiHoc 2003, 2003, pp. 1–12.
[8] K. Ramamritham and C. Pu, A formal characterization of Epsilon serializability, *IEEE Transactions on Knowledge and Data Eng* **7**(6) (1995), 997–1007.
[9] M. Ikeda, L. Barolli, G.D. Marco, T. Yang, A. Durresi and F. Xhafa, Tools for performance assessment of OLSR protocol, *Mobile Information Systems* **5**(2) (2009), 165–176.
[10] N. Krishnakumar and R. Jain, Escrow techniques for mobile sales and inventory applications, *Wireless Networks* **3**(3) (1997), 235–246.
[11] N. Preguica, C. Baquero, F. Moura, J.L. Martins, R. Oliveira, H. Domingos, J.O. Pereira and S. Duarte, *Mobile Transaction Management in Mobisnap*, In Proceedings of ADBIS-DASFAA 2000, 2000, pp. 379–386.
[12] P. Padmanabhan, L. Gruenwald, A. Vallur and M. Atiquzzaman, A survey of data replication techniques for mobile ad hoc network databases, *VLDB Journal* **17**(5) (2008), 1143–1164.
[13] P.E. O'Neil, The Escrow transactional method, *ACM Trans. on Database Systems* **11**(4) (1986), 405–430.
[14] R. Alonso, D. Barbara, and H. Garcia-Molina, Data caching issues in an information retrieval system, *ACM Transactions on Database Systems* **15**(3) (1990), 359–384.
[15] S.K. Madria, Timestamps to detect R-W conflicts in mobile computing, In Proceedings of International Workshop on Mobile Data Access (MDA'98), 1998, pp. 242–253.
[16] S.S. Manvi, M.S. Kakkasageri and J. Pitt, Multiagent based information dissemination in vehicular ad hoc networks, *Mobile Information Systems* **5**(4) (2009), 363–389.
[17] T. Hara, *Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility*, In Proceedings of IEEE Infocom 2001, 2001, pp. 1568–1576.
[18] T. Hara, Replica allocation methods in ad hoc networks with data update, *ACM-Kluwer Journal on Mobile Networks and Applications* **8**(4) (2003), 343–354.
[19] T. Hara and S.K Madria, Data replication for improving data accessibility in ad hoc networks, *IEEE Transactions on Mobile Computing* **5**(11) (2006), 1515–1532.
[20] T. Hara and S.K Madria. Consistency management strategies for data replication in mobile ad hoc networks, *IEEE Transactions on Mobile Computing* **8**(7) (2009), 950–967.
[21] T. Hara, *Escrow Approaches for Global Consistency in Mobile Ad Hoc Networks*, In Proceedings of Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009), 2009, pp. 343–350.
[22] Y. Huang, P. Sistla and O. Wolfson, *Data Replication for Mobile Computer*, In Proceedings of ACM SIGMOD'94, 1994, pp. 13–24.

Takahiro Hara received the B.E, M.E, and Dr.E. degrees from Osaka University, Osaka, Japan, in 1995, 1997, and 2000, respectively. Currently, he is an Associate Professor of the Department of Multimedia Engineering, Osaka University. He has published more than 100 international Journal and conference papers in the areas of databases, mobile computing, peer-to-peer systems, WWW, and wireless networking. He served and is serving as a Program Chair of IEEE International Conference on Mobile Data Management (MDM'06 and 10) and IEEE International Conference on Advanced Information Networking and Applications (AINA'09). He guest edited IEEE Journal on Selected Areas in Communications, Sp. Issues on Peer-to-Peer Communications and Applications. He was a PC Vice-chair of IEEE ICDE'05, IEEE ICPADS'05, IEEE NBiS'09, CSA-09, and IEEE AINA'06, 07, 08, and 10. He served and is serving as PC member of more than 100 international conferences such as IEEE ICNP, WWW, DASFAA, ACM MobiHoc, and ACM SAC. His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is an IEEE Senior member and a member of four other learned societies including ACM.

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