01 Jul 2009

Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks

Takahiro Hara

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Recommended Citation

T. Hara and S. K. Madria, "Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing (TMC)*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2009.
The definitive version is available at https://doi.org/10.1109/TMC.2008.150

Consistency Management Strategies for Data Replication in Mobile Ad Hoc Networks

Takahiro Hara, *Senior Member*, *IEEE*, and Sanjay Kumar Madria, *Senior Member*, *IEEE*

Abstract—In a mobile ad hoc network, data replication drastically improves data availability. However, since mobile hosts' mobility causes frequent network partitioning, consistency management of data operations on replicas becomes a crucial issue. In such an environment, the global consistency of data operations on replicas is not desirable by many applications. Thus, new consistency maintenance based on local conditions such as location and time need to be investigated. This paper attempts to classify different consistency levels according to requirements from applications and provides protocols to realize them. We report simulation results to investigate the characteristics of these consistency protocols in a mobile ad hoc network.

Index Terms—Mobile ad hoc networks, consistency management, data replication, mobile computing.

1 Introduction

In *mobile ad hoc network (MANET)* [14], 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 [9], [15]. To improve data availability, data replication is the most promising solution [4], [8]. Based on this idea, we have designed effective data replication techniques in MANETs in our previous papers [9], [10], [12].

In [10] and [12], we assume that replicas of a data item become invalid after the host holding the original updates it and the consistency of data operations on replicas is kept in the entire network. However, since network partitioning frequently occurs in a MANET, this strong consistency management scheme heavily deteriorates the data availability. Moreover, many applications in MANETs do not require such a strong consistency. For instance, consider a situation where members of a rescue service that constructs a MANET in the disaster area are divided into several groups each of which is responsible of a certain region and the members in each group share various kinds of information such as that on the extent of damages. In this situation, the consistency of data operations to data items that are used locally in each group must be strictly kept in the same group and is not required to be maintained strictly in different groups.

In this paper, we discuss different consistency conditions of data operations on replicas in MANETs. First, we classify consistency levels according to application requirements. Next, we propose protocols to achieve them and, then, discuss the impact of replica allocation for the system performance when the memory space of mobile hosts is limited. We also report simulation results to investigate the behavior of the proposed protocols.

It should be noted that our proposed consistency levels and protocols for achieving them are not very novel because these are basically common and simple approaches to maintain the consistency based on a typical quorum system and time-based coherency condition. The main contributions of this paper are not only the proposal of the consistency levels and protocols but 1) the classification of consistency levels according to the system and application requirements, 2) the choices of the existing techniques and their extensions for design of the protocols of these consistency levels in MANETs, and 3) performance studies of these protocols.

Note that some of the results of this paper have been reported in [11].

2 Related Work

Consistency management is a popular research topic in distributed database systems. For example, Alonso et al. [1] discussed cache coherency issues and classified several coherency conditions such as time-based, value-based, and version-based ones. As mentioned in Section 1, our proposed consistency levels are basically based on conventional approaches. Time-based Consistency (TC) in this paper is similar to the time-based coherency condition, *default coherency condition*, in [1]. 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 [21]. Some of these conventional approaches are applicable to our proposed consistency levels, which is open to our future work.

In mobile environments (not MANET), several consistency management strategies that consider host disconnections have been proposed [13], [17], [19], [20]. Most of them

- *T. Hara is with the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan. E-mail: hara@ist.osaka-u.ac.jp.*
- *S.K. Madria is with the Department of Computer Science, Missouri University of Science and Technology, 500 West 15th Street, 325 Computer Science Building, Rolla, MO 65409. E-mail: madrias@mst.edu.*

Manuscript received 25 Dec. 2006; revised 24 Oct. 2007; accepted 12 Apr. 2008; published online 15 Oct. 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0344-1206.
Digital Object Identifier no. 10.1109/TMC.2008.150.

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 on original data and its replicas with low communication costs. These strategies assume only one-hop wireless communication.

In [19], the authors proposed the concept of *d-consistency*, which is a cluster-based approach and allows a certain degree of divergence of values of copies in different clusters. Similar to Epsilon serializability [1], this concept is also applicable to some of our proposed consistency levels.

Several methods have been proposed for preserving consistency of data operations on replicas in MANETs [15], [16], [22]. In [15], the authors proposed methods in which the consistency is maintained by employing a strategy based on the *quorum system* that has been proposed for distributed databases [2]. In [16], the authors extended the methods proposed in [15] by applying *probabilistic quorum system* [18]. Their methods are considered similar to ours because consistency of data operations is maintained based on the quorum system. However, in [15] and [16], the authors aimed to roughly keep the consistency in the entire network. Therefore, the locality in MANETs was not taken into account.

In [22], the authors defined two different consistency levels, *local observation consistency* and *global observation consistency*. Global observation consistency is equivalent to Global Consistency (GC) in this paper. Local observation consistency is almost equivalent to Peer-based Consistency (PC), except that it requires replicas to eventually converge to the most recent version. In [22], only two different consistency levels are defined, whereas in here we define five levels. Moreover, the authors tried to keep consistency based on an optimistic manner, i.e., transactions are tentatively committed and the consistency is checked by using serializability graphs. Such an optimistic approach may not work well in MANETs due to frequent conflicts of data operations performed in partitioned networks.

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 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 given as follows:

- Each mobile host (peer) knows its current location by using some device such as GPS and moves around in the given area. Peers communicate with others using wireless communication. 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.
- The MANET consists of 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 does not encounter a node failure. (Note that we can easily remove these assumptions by adopting some existing techniques for turning over the role of the proxy to another 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 where peers can move across regions. In this paper, we basically assume the former case but also discuss how to deal with the latter case.
- Each proxy and peer knows all proxies in the entire network. This assumption is easy to find in many real situations. In an example of rescue service, it is natural that every member knows group leaders who act as proxies. Even when members do not know each other, a new peer has to register its participation to the proxy to join the MANET, and the peer can get the information on all proxies from the proxy. Here, every proxy can know all the others at the configuration phase of the MANET.
- The set of all regions in the entire network is denoted by $R = \{R_1, R_2, \ldots, R_l\}$, where l is the total number of regions and R_i $(i = 1, \ldots, l)$ 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. For example, in an environmental sensing operation where the entire area has to be examined uniformly, the area should be divided into shared-nothing regions.
- If proxies are not within direct communication range of their neighboring proxies, communication packets are forwarded via other peers in a multihop manner.
- Data are handled as a collection of data items. We assign a unique *data identifier* to each data item located in the system. The set of all data items is denoted by $D = \{D_1, D_2, \ldots, D_n\}$, where n is the total number of data items and D_j $(1 \leq j \leq n)$ is a data identifier.
- Each peer performs read and write (update) operations to any data items. For simplicity, we basically assume blind writes, where a peer writes a value without reading the latest value before. We can easily remove this assumption in our protocols to achieve GC, Local Consistency (LC), and PC by appropriately setting some system parameters, e.g., quorum sizes.
- We assume a simple transaction model in which each transaction consists of a single database operation (read or write). Thus, the consistency of data operations on replicas is defined such that every read operation reads a valid replica. Here, a valid replica is the latest version in a specific area (the area is different according to the consistency

levels) except for TC in which a replica whose version is within a predetermined time from its reading time is considered valid. 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 multiple data operations. We do not consider such complex transactions in our work.

Here, our model allows different versions of replicas of each data item under the condition that every read operation reads a valid replica in a specific area. This is different from a problem of maintaining replica coherency in which replicas necessarily converge on one version.

- We assume 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. Some conventional works [15], [16] made this assumption for the purpose of simplicity. When data items of small sizes such as location and statistical data are shared for collaborative work, this assumption is reasonable.

  The second case is where only proxies have unlimited memory space for replicating all data items and other peers have no memory space. This is a typical case where proxies have much more resources and computational power than other peers. A good example is a rescue or military applications where group leaders (proxies) are equipped with powerful mobile machines and other members (peers) are equipped with small mobile devices.

  The last case is where proxies and peers have limited memory space for replication. This is a more general assumption depicting a real situation. Here, we do not restrict to any particular replication strategy to allocate replicas. It is assumed that the proxy knows replicas held by peers in its region. This is achieved by sending the information on replicas held by a peer to the proxy when the peer participates in a new region.

4 Classification of Consistency Levels

Since there are various applications in MANETs such as information sharing in a rescue service and distributed data processing in sensor networks, there cannot be one universal optimal strategy for consistency management. Thus, in this section, we propose four different primitive consistency levels, GC, LC, TC, and PC and one combined consistency level. Here, which consistency level an application requires is determined based on for what the data obtained by read operations are used in the application and for whom (individual, group, or everyone) the data are useful. Thus, in this section, we describe the requirement for read operations in each consistency level. Then, in the next section, we describe how to handle read and write operations to meet these requirements.

4.1 Global Consistency (GC)

The consistency of data operations on replicas is required in the entire network. This is equivalent to the traditional notion of GC. 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 that was written by the latest write operation issued in the entire network. Providing such a strong consistency requires many hops of message passing and, therefore, is hard to achieve in MANETs. Also, many applications do not desire it.

However, there exist some applications that require GC in MANETs. A good example is a situation in which members of a rescue service are divided into several groups each of which is responsible of a certain region and the information on the progress of tasks assigned to each group 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 machine or human resources, and scheduling of new tasks. Thus, the consistency of data operations must be maintained strictly in the entire network.

4.2 Local Consistency (LC)

The consistency of data operations on replicas is required only in each region of interest. 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 that 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 share the information on the damages such as the number of injured persons and destroyed buildings, which can be separate data items according to the extent of the damages. This information is used locally by the leader in each region to decide the resource allocation and task scheduling in the group. Since this information is referenced only by the leader and members in the same group, it is not necessary and too costly to keep the strict consistency in the entire network. Therefore, in this case, LC is suitable.

4.3 Time-Based Consistency (TC)

In TC, replicas are valid even if their versions are different but have not passed a predetermined time (validity period T) since they have been updated last. This consistency level weakens the strictness of consistency from the temporal perspective.

Specifically, when the version of a replica of data item D_j is V_j, i.e., the latest write operation on the replica was performed at time V_j, a read operation on the replica succeeds if the read operation is issued at time A_j on condition that $V_j + T > A_j$. Here, we assume peers use some logical clocks and time is synchronized among all peers by applying some conventional protocols such as [7].

TC can be usually applied to data items whose values continuously (preferably slowly) change and the values themselves are not very important for the application. A good example is a situation in which members in a rescue

service share the information on locations of the members. This information can be used by the members for various purposes such as temporarily constructing a subgroup of nearby members to carry out some tasks, e.g., removing a heavy rock.

4.4 Peer-Based Consistency (PC)

The consistency of data operations is required only in each peer. Thus, this consistency level further weakens the strictness of consistency than LC and is the weakest from the spatial perspective. Formally, PC requires that at each peer, every read operation issued by the 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, e.g., check out and describe features of damaged buildings, where the survey result (review) of each object is recorded as a data item. In this case, a member engaged in the survey can read and update his own review on each object anytime, independent of reviews of others on the same object.

4.5 Application-Based Consistency (AC)

Since different applications running in a MANET might require different consistency levels, hybrid consistency levels should be considered. In Application-Based Consistency (AC), assuming that there exist different kinds of applications requiring different consistency levels, all the required consistency levels are satisfied.

Here, applications running in a MANET might require just a subset of the four primitive consistency levels rather than all, thus, AC is an extreme case. Since how to choose and design appropriate combinations of different consistency levels is not a focus in terms of the scope of this paper, we just introduce AC as an example of such hybrid approaches.

Note that to achieve AC, write operations must be handled to provide the strongest consistency level, i.e., GC, while read operations can be handled in different ways according to the application requirements, i.e., the way to provide just the same consistency level as the application requires.

5 Consistency Management Protocols

In this section, we describe protocols to realize the five consistency levels given in the previous section.

As for the three primitive consistency levels except for TC, the system has to guarantee peers to read the latest version in a specific area, i.e., in the entire network for GC, in the region for LC, and in the peer itself for PC. In doing so, we need protocols to control write and read operations because the latest write operation performed on each data item must be distinguished in the specific area.

5.1 Global Consistency (GC)

5.1.1 Basic Idea

To realize GC, the simplest way is "read-one write-all," which is a traditional and common approach to maintain consistency among data operations on replicas. However, since "read-one write-all" requires that every write operation is performed on all the replicas, it works badly in MANETs due to frequent peer disconnections and network partitioning, i.e., most write operations fail. Alternatively, we consider a quorum system following [15] and [16].

In a quorum system, read and write operations are performed on only replicas held by mobile hosts that form read and write quorums, respectively, where every pair of read and write quorums have an intersection. Specifically, when a mobile host updates a data item, it performs the write operation on replicas held by all mobile hosts in a write quorum so that replicas of the latest version exist in the quorum. On the other hand, when a mobile host reads a data item, it performs the read operation on replicas held by all mobile hosts in a read quorum. 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, and thus, the consistency of data operations 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 mobile hosts that hold replicas disconnect from the network or network partitioning occurs.

5.1.2 Overview

We employ a quorum system based on dynamic quorums similar to [16], 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 MANETs, it is not wise to adopt a strategy that requires complex calculation and a large amount of information for the calculation. Thus, we adopt a very simple strategy that calculates the sizes of quorums and dynamically constructs quorums with the calculated sizes, by using the information that is easily available by proxies. Specifically, to calculate quorum sizes, each proxy uses the information on the total number of regions (proxies) in the entire network and on the total number of peers having each replica in its responsible region. In the following, we explain the details.

First, the quorum size for a write operation (to any data item), $|QW|$, and that for a 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_i $(i = 1, \ldots, l)$, the quorum size for a write operation on data item D_j, $|QLW_{ij}|$, and that for a read operation, $|QLR_{ij}|$, are determined where the condition $|QLW_{ij}| + |QLR_{ij}| > P_{ij}$ is satisfied. Here, P_{ij} is the total number of peers that hold D_j in the region. If P_{ij} is 1, both $|QLW_{ij}|$ and $|QLR_{ij}|$ 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 $|QLW_{ij}| + |QLR_{ij}| > P_{ij}$, 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 $|QLR_{ij}|$ ($|QLW_{ij}|$) replicas.

Here, in our protocol, to dynamically determine a local quorum, the proxy has to maintain the information on the sizes of the local quorums, $|QLW_{ij}|$ and $|QLR_{ij}|$, for each data item. In distributed database systems, various kinds of metadata such as information on data creator, version (time stamp), size, and keywords are attached to each data item. Thus, the information on the local quorum sizes could also be attached to each data item as metadata. Since the size of the information on the quorum sizes is very small, e.g., 1 byte for each, it does not raise a scalability problem even when granularity of the data items is fine.

5.1.3 Protocol

In this clause, we present the protocol to achieve GC. Since GC requires hierarchical data operations in the entire network, it takes long time, e.g., a few seconds or more, to perform. Thus, to cope with network topology change and peer failures during the protocol execution, we adopt a three-phase approach for a write (read) operation, which basically consists of three rounds of message exchanges, 1) a lock request/its reply, 2) write operations/its acknowledgment (a request for the latest replica/data transfer), and 3) a message for commit and lock release/its acknowledgment (acknowledgment and lock release).

In the following, we describe the details of each of the three phases. For easy understanding, we first assume the limited movement case, where peers do not move beyond their existing regions. Then, in the next clause, we discuss how to deal with the unlimited movement case for peers.

Phase 1. When a read (write) operation is issued by a peer in region R_i to data item D_j, first, the peer unicasts a request (query) for the operation to the proxy of the region. If the peer does not connect to the proxy, the peer tries to find another proxy in a different region by sequentially unicasting the request to proxies (from those in closer regions) until the request reaches a proxy. If the peer fails to find a proxy, the operation fails immediately. Otherwise, the proxy that received the request becomes the *coordinator* to process the operation.

The coordinator tries to set *global read (write) locks* to arbitrary $|QR|$ ($|QW|$) replicas held by proxies including itself (*Global quorum*). This is done by sequentially unicasting a global lock request to other proxies (from those in closer regions) until the total number of the global locks set on replicas reaches $|QR|$ ($|QW|$). Note that a global lock is not an actual lock for data operation but a virtual lock, which represents whether or not the necessary number of local locks are set on replicas held by peers in the region as explained later.

Each proxy in region R_i that received the request tries to set *local read (write) locks* to arbitrary $|QLR_{ij}|$ ($|QLW_{ij}|$) replicas of data item D_j held by itself and peers in the region (*Local quorum*). If $|QLR_{ij}|$ ($|QLW_{ij}|$) is 1 and the proxy holds D_j, the proxy sets the local lock to its holding replicas. Otherwise, the proxy multicasts a local lock request to all peers that hold D_j in the 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 proxy. In the case of a read operation, the reply contains the information on the version of the replica.

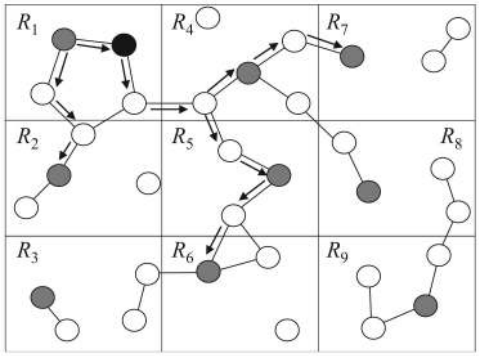

Fig. 1. Example of executing GC.

If the proxy succeeds to set equal or more than $|QLR_{ij}|$ ($|QLW_{ij}|$) local locks, the global read (write) lock is set on the replica that the proxy holds and the proxy sends back a reply to notify the coordinator of the fact. In the case of a read operation, the reply contains the information on the version of the latest replica in the region. In addition, the proxy records the IDs of peers that replied to the request and, in the case of a read operation, the IDs of peers that hold the latest replica. Otherwise, if the proxy failed to set the necessary number of local locks, it also sends back a reply to notify the coordinator of the fact.

If the coordinator succeeds to set $|QR|$ ($|QW|$) global locks, it sends a query reply to notify the request-issuing peer of the fact. The reply contains the IDs of proxies with the global lock, and in the case of a read operation, the IDs of proxies having the latest replica in the entire region and its version. In the case of a write operation, the coordinator notifies the request-issuing peer of the success of phase 1 and requests the replica of the new (latest) version. Then, the procedure goes to phase 2. Otherwise, the operation request fails immediately and the coordinator notifies the request-issuing peer of the fact.

Fig. 1 shows an example of executing phase 1 of this protocol. In this example, there are nine shared-nothing regions $(R_1, \ldots, R_9)$. A circle denotes a peer, whereas a gray one denotes a proxy. A solid line between two peers denotes a wireless link. Here, let us suppose that every peer has enough memory space to replicate all data items. Thus, P_{ij}, the number of peers having a replica of data item D_j in region R_i, is equal to the total number of peers in R_i for all data items, i.e., $\{P_{1,j}, P_{2,j}, P_{3,j}, P_{4,j}, P_{5,j}, P_{6,j}, P_{7,j}, P_{8,j}, P_{9,j}\} = \{4, 4, 4, 5, 3, 3, 3, 4, 4\}$. Let us also suppose that $|QR| = 5$, $|QW| = 5$, $|QLR_{ij}| = \lfloor P_{ij}/2 \rfloor$, and $|QLW_{ij}| = P_{ij} - |QLR_{ij}| + 1$.

Here, we assume that the black colored peer in R_1 issues a write operation on D_1. The proxy in the region becomes the coordinator and tries to set a global lock to $|QW|$ $(= 5)$ proxies from those in the closer regions to farther ones, i.e., the order is $\{R_1\}$, $\{R_2, R_4\}$, $\{R_3, R_5, R_7\}$, $\{R_6, R_8\}$, and $\{R_9\}$, if we use the Manhattan distance. Among the proxies in six regions whose

Manhattan distance is equal or less than 2, the coordinator succeeds to set a global lock to four proxies (R_1, R_2, R_4, R_5) because it has a communication path (denoted by directional arrows) to each of them, and these proxies connect to an equal or larger number of peers in their region than the local quorum sizes. Specifically, $\{|QLW_{1,1}|, |QLW_{2,1}|, |QLW_{4,1}|, |QLW_{5,1}|\} = \{3, 3, 3, 2\}$, and the numbers of accessible peers from the proxies are {4, 3, 4, 3}.

Then, R_1 tries another proxy in the region whose Manhattan distance is 3 (R_6 or R_8). Here, it succeeds to set a global lock to the proxy in R_6. Since the total number of proxies with a global lock becomes 6, phase 1 successfully completes.

Phase 2: read operation. In this phase, the requested data operation is performed on replicas with locks. As for read operations, if the request-issuing peer holds the latest version (it can be known from the information on the latest version attached with the query reply), it performs the read operation on its own replica. In this case, the coordinator does nothing and the procedure goes to the next phase. Otherwise, the coordinator unicasts a data transmission request to the closest proxy having the latest version in its region.

The proxy that received the data transmission request sends a local data transmission request to the closest peer that holds the latest replica. The peer that received the request transfers its holding replica to the proxy. Then, the proxy sends back the acknowledgment to the sender peer and forwards the replica to the coordinator. The coordinator that received the latest replica sends back the acknowledgment to the sender proxy and forwards the replica to the request-issuing peer. Then, the procedure goes to phase 3.

Note that it can happen that the proxy that holds the latest replica becomes inaccessible from the coordinator or all the peers having the latest replica in the region become inaccessible from the proxy during the execution of phase 2. In such a case, the coordinator tries to find another latest replica holder. Here, the coordinator knows the proxies having the latest version in their regions, and these proxies also know peers having the latest version. Thus, from the closest proxy to farther ones, the coordinator successively sends a data transmission request to the proxy until it obtains a latest replica. If all these procedures fail, the coordinator aborts the request. Of course, if the request-issuing peer becomes inaccessible from the coordinator, the request fails immediately.

Phase 3: read operation. In this phase, the request-issuing peer that received a latest replica sends an acknowledgment to the coordinator, and then the coordinator sends a commit and lock release message to the proxies involved in the global quorum. Each proxy that received this message forwards it to the peers involved in the local quorum in the region. Then, all the locks set to replicas for this operation are released and the procedure finishes. Even if the commit and lock release message cannot reach the proxy and peers involved in the quorums, it does not affect the database state, thus, the coordinator can complete the operation.

Phase 2: write operation. In phase 2 of the write operation, the coordinator receives the replica of the new version from the request-issuing peer and forwards it to $|QW|$ proxies with the global locks.

Each proxy that received the replica forwards it to the closest $|QLW_{ij}|$ peers having replicas with the local lock. Each peer that received the replica of the new version replaces its holding replica with the received one and sends back the acknowledgment to the proxy. If the proxy receives the acknowledgment from all the $|QLW_{ij}|$ peers, it sends back the (positive) acknowledgment to the coordinator. If the coordinator receives the acknowledgment from all the $|QW|$ proxies, the procedure goes to phase 3.

Here, it can happen that only less than $|QLW_{ij}|$ peers having locked replicas are accessible from the proxy or some proxies with the global lock become inaccessible. In such a case, if the missing peers are proxies, finding alternative proxies that can set the necessary number of local locks is very costly, thus, in our protocol the coordinator aborts the request. On the other hand, if the missing ones are peers, the proxy in the region with the missing peers tries to fine-adjust the quorum by searching alternative accessible peers in the region. The overhead of this process is low because the proxy can reuse the result of phase 1, i.e., the information on accessible peers. If it fails, the coordinator aborts the request.

Phase 3: write operation. In phase 3 of the write operation, the coordinator sends a commit and lock release message to the proxies involved in the global quorum. Each proxy that received this message forwards it to the peers having replicas with the local lock. Then, each peer that received the message sends back the acknowledgment to the proxy and releases the local lock set to its holding replica. If the proxy receives the acknowledgment from all the peers involved in the local quorum, it sends back the acknowledgment to the coordinator and releases the global lock. If the coordinator receives the acknowledgment from all the proxies involved in the global quorum, it sends a message notifying the success of the operation to the request-issuing peer and the procedure finishes.

If phase 3 fails during the execution and the missing peers are not proxies, the coordinator tries to find alternative available peers in the same way as the fail in phase 2, and then, phase 3 is executed again to replicas that are newly performed by the operation. If it also fails, the operation fails and the coordinator sends an abort message to all the proxies and peers having locked replicas.

Here, to ensure the execution of write operations on replicas held by multiple nodes in a distributed system, *two-phase commit (2PC)* is the most popular approach. 2PC achieves this by two rounds of message transmissions between the coordinator and all the replica holders; *commit prepare/acknowledgment* and *commit/acknowledgment*. However, the overhead of two rounds of message transmissions is very high for MANETs. To solve this problem, there are several possible approaches. For example, we can omit the first round (commit prepare/acknowledgment) by merging it with phase 2 of our protocol. This is possible because in our protocol, phase 2 is basically performed synchronously among peers having replicas with locks and requires the acknowledgment. Also, if the underlying routing protocol requires flooding of a route request before sending

communication packets, the messages of the first phase in 2PC can be piggybacked on the route request and its reply. Since the possible approaches depend on the implementation of the underlying network protocols and other factors, we do not restrict a particular one in our protocol.

Time-out in processing. Each of the three phases 1, 2, and 3 has the time-out at both proxy and peer. As for proxy time-out, if the proxy cannot receive the required number of replies within the time-out, it retries to find other ones. As for peer time-out, if a peer has not received a message or request for the next procedure after it replied to the proxy for the previous phase, it aborts all the procedures performed for the corresponding operation.

5.1.4 How to Deal with Peers' Movement across Regions

The protocol of GC (also LC described later) uses the number of replica holders in each region to construct local quorums. Here, the number of replica holders in each region, P_{ij}, dynamically changes in the case of unlimited movement where peers move across regions. Therefore, a mechanism to maintain the members (peers) in each group is needed.

In this clause, we first explain a mechanism for member maintenance. Then, we present how our protocol works in the case of unlimited peer movement. Here, all the procedures except for that for calculating the quorum size are basically the same as those of the original protocol described above.

Mechanism for member maintenance. When a peer moves into a new region, the peer tries to notify the proxy in the region (where the peer previously existed) of its exit from the region by sending its peer identifier. When the peer cannot make this, i.e., it does not connect to the proxy, the peer is considered in the region even if it is out of the region. If the peer succeeds in the exit notification, it then notifies the proxy in the new region of its entrance by registering its peer identifier. When the peer cannot make this registration, the peer is considered not in the region even if it does exist geographically in the region. In this case, the peer is considered in no region. Here, peers that failed to register exit or entrance retry it periodically.

Extension of the GC protocol. As for write operations, the local quorum in the region can be dynamically constructed based on the current value of P_{ij}. As for read operations, it is not enough to consider the current value of P_{ij}. Instead, 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'_{ij} that hold D_j in the region at T'_j. Since, in the protocol of GC, every read and write request is processed through the proxy in the region, this extension is easy to achieve.

Based on the above extension, each proxy can calculate the local quorum size for a read operation in the region. Specifically, by using the recorded information on P'_{ij}, the proxy can calculate $|QLW'_{ij}|$, which is the local quorum size for the latest write operation. Then, the quorum size for a read operation in the region is initially set as $|QLR_{ij}| = P'_{ij} - |QLW'_{ij}| + 1$ to guarantee that every pair of local write and read quorums has an intersection. After then, $|QLR_{ij}|$ is decreased by one every time 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, the time tp when the peer came into the region, and the version of the replica of D_j held by the peer.

When a read operation to D_j is issued by a peer, the proxy in the region sends a local lock request to all peers that have a replica of D_j and have belonged to the region before and after the latest write time, i.e., $T'_j > tp$.

Theorem. *The extended protocol can preserve GC in environments where peers move across regions.*

Proof. First, we focus on consistency inside a region. When a write operation is performed on data item D_j at time T'_j, replicas of D_j in the region can be categorized into two different classes; $|QLW'_{ij}|$ replicas on which the latest write operation was performed and $P'_{ij} - |QLW'_{ij}|$ replicas on which the operation was not performed.

Then, we assume that a read operation is issued in the region at time T''_j ($T''_j > T'_j$). We also assume that among the P'_{ij} peers, E_j peers have exited from the region during the time interval from T'_j to T''_j, among which E'_j peers hold old (not the latest) replicas. In this case, there are still $|QLW'_{ij}| - (E_j - E'_j)$ latest replicas and $P'_{ij} - |QLW'_{ij}| - E'_j$ old replicas in the region, all of which were written in the region. Since in the extended protocol, the quorum size for the read operation is set as $|QLR_{ij}| = P'_{ij} - |QLW'_{ij}| - E'_j + 1$ and only peers that have belonged to the region before and after the latest write operation was issued are involved in the read quorum, an arbitrary read quorum contains at least one latest replica (because the total number of old replicas in the region is $P'_{ij} - |QLW'_{ij}| - E'_j$). Thus, the consistency of data operations is preserved in the region.

Since the consistency is preserved in the region, the consistency in the entire network is also preserved by using the global quorum described in Section 5.1.3. □

5.1.5 Correction of Message Arrival Order

In a real environment, due to the delay of message propagation on a multihop route, two messages generated from two different peers are often received by other peers in the reverse order. This may cause a deadlock of requests that try to set locks to quorums. In this paper, a peer that received request messages can fix the order even when their arrival times are reversed, by using the information on the query issued time that is attached to every request message (query, local lock, and global lock).

5.2 Local Consistency (LC)

5.2.1 Overview

Similar to GC, we employ dynamic quorums to realize LC. Since consistency of data operations on replicas is managed only among peers in each region, LC allows different versions of replicas in different regions; thus, reconciliation to converge the values of replicas is not necessary.

The procedure of the protocol to achieve LC is basically the same as the procedure in the region of the GC protocol, which also consists of three rounds of message exchanges. Thus, we briefly explain it and omit the details.

In each region R_i $(i = 1, \ldots, l)$, the quorum size for a write operation to data item D_j, $|QLW_{ij}|$, and that for a read operation, $|QLR_{ij}|$, are determined where the condition $|QLW_{ij}| + |QLR_{ij}| > P_{ij}$ is satisfied. If P_{ij} is 1, both $|QLW_{ij}|$ and $|QLR_{ij}|$ are set to 1. Owing to condition $|QLW_{ij}| + |QLR_{ij}| > P_{ij}$, it is guaranteed that every read operation reads a replica of the latest version in the region.

5.2.2 Protocol

When a read (write) operation is issued by a peer in region R_i, the peer tries to set read (write) locks to arbitrary $|QLR_{ij}|$ ($|QLW_{ij}|$) replicas held by the proxy and peers in the region in which the peer exist (*Local quorum*). If equal to or more than $|QLR_{ij}|$ ($|QLW_{ij}|$) peers or the proxy replied, the read (write) operation is performed on the replicas with the locks. As for read operations, the operation is done on a replica of the latest version among those with locks. As for write operations, the operation is performed on the closest $|QLW_{ij}|$ replicas held by peers in the quorums.

We show some examples of executing phase 1 of the LC protocol using Fig. 1. When a peer in R_9 issues a write operation on D_1, phase 1 of the LC protocol succeeds because every peer in R_9 connects to more than $|QLW_{9,1}|$ $(= 3)$ peers. Here, if the required consistency level is GC, this request fails because the proxy in R_9 cannot set the necessary number of global locks $(|QW| = 5)$.

5.2.3 How to Deal with Peers' Movement across Regions

The protocol of LC can be also extended to deal with peers' movement across regions in the same way as that of GC. However, to do so, the protocol of LC has to be changed to make every read and write request be processed through the proxy in the region. This has a shortcoming that the flexibility of choosing a quorum is slightly decreased because request-issuing peers have to connect to the proxy in the region.

5.3 Time-Based Consistency (TC)

5.3.1 Overview

TC does not require peers to read the latest version but requires them to read replicas whose version is within the valid period T. Therefore, as for TC, validity of a replica is affected only by the last write operation that was performed on the replica but not affected by any write operations performed on other replicas. Namely, the protocol does not need to control write operations in terms of consistency, i.e., write operations can be performed on any replicas either valid or invalid. On the other hand, read operations need to find a valid replica. If the peer that issued a read operation request holds a valid replica, it can perform the operation locally.

5.3.2 Protocol

When a write operation to a data item is issued by a peer and the peer holds a replica of the target item, the operation is performed on the replica. If the peer does not hold the replica, it tries to find a replica held by a peer in its region. This is done by broadcasting the write request in the region because peers do not know the members in the same region. If the peer receives replies from peers having a replica, the write operation is performed on a replica held by the closest peer. If it fails, the peer tries to find a replica held by a peer in another region by broadcasting the request in the entire network. If it succeeds, the operation is performed on a replica held by the closest peer.

As for read operations, the same procedure as a write operation is performed except that the peer has to find a valid replica. Specifically, the information on the read operation issued at time A_j is attached to the request message sent to other peers. Then, at each peer that received the request, it is checked whether condition $V_j + T > A_j$ is satisfied for its holding replica, where V_j is the version of the replica. If the replica is valid, the reply is sent to the request-issuing peer.

5.3.3 Discussions

As described above, in terms of consistency, a write operation can be performed on any replica, thus, we adopt a simple approach. However, in terms of performance such as transaction success ratio and communication overhead, there are various other choices. For example, to improve read success ratio, a write operation performed on a peer is better to be propagated to distribute newer replicas to other peers, which can be done based on lazy updates. Here, the propagation of write operations is not good in terms of communication overhead. Thus, there is a trade-off relationship between read success ratio and communication overhead in TC.

5.4 Peer-Based Consistency (PC)

PC just requires each peer to read the latest version written by it. Thus, PC can be applied only to replicas that the peer holds. In other words, basically, PC is defined only in cases where every peer has enough memory space to replicate all data items that the peer accesses.

In our assumed system model, since every transaction consists of a single operation, each peer can perform read and write operations anytime on its own replica and no special mechanism is needed. Namely, database operations issued at a peer do not affect those issued at other peers, i.e., PC allows different versions of replicas. Thus, reconciliation to converge the values of replicas is not necessary.

5.5 Application-Based Consistency (AC)

AC provides all consistency levels required from different applications. Therefore, in the AC protocol, write operations are performed in the same way as the GC protocol. On the other hand, read operations are performed in different ways according to the consistency levels that the request-issuing peers require. For example, if the request-issuing peer requires LC for a read operation, the request is processed by the LC protocol, and so on.

5.6 Impact of Replica Allocation

If proxies and peers have limited memory space, they have to determine which data items to replicate. In such a

situation, the performance for consistency management is affected by the replication strategy. More specifically, in GC and LC, the quorum size for a write (read) operation in the region $|QLW_{ij}|$ ($|QLR_{ij}|$) changes according to the number of replicas of each data item. This directly affects the communication cost and might affect the data availability. Moreover, the number of replicas also affects the hopcount to the closest replica holder in read operation, i.e., more replicas generally result in a shorter hopcount. In the extreme case that a region has only one copy of an item, the traffic for constructing a quorum is very low. However, it makes the hopcount for reading the replica longer, becomes a single point of failure, and unbalances load and power consumption among peers. This has a large impact on performance when the read frequency is much higher than the write frequency and the sizes of data items are large.

Similarly, in TC, the number of replicas affects the communication costs and the data availability because a request-issuing peer has to search another peer that holds a (valid) replica of the target data item if it does not hold.

Therefore, in the simulation experiments, we examined the impact of two typical replication strategies; 1) *square-root* allocation (SRA)* and 2) *inversed SRA (ISRA)*. SRA is a popular strategy used in an unstructured P2P network [5], in which the ratios of numbers of replicas are proportional to the square-root of the query (read) frequencies to the data items. It is known that SRA achieves the optimal replica allocation in terms of query efficiency, i.e., it is effective for finding one replica in the entire system. However, it is not always effective for a quorum system because it requires many message transmissions to construct a quorum and also many data transmissions for a write operation. On the contrary, ISRA allocates replicas, in which the ratios of numbers of replicas are inversely proportional to the square root of the read frequencies to the data items. This strategy is effective to reduce message transmissions to construct a quorum for hot data items.

Here, it can happen that there is no copy of a certain data item in some region if the adopted replication strategy is not aware of the number of replicas of each data item in each region. This might cause considerable degrading in data availability. Thus, in a real environment, some strategies should be considered to keep at least one copy of each data item in every region. Since such a strategy is beyond the scope of our paper, we assume the above typical strategies to fairly verify the impact of the number of replicas on the performance.

6 Simulation

In this section, we show simulation results to investigate the characteristics of the proposed consistency levels and protocols. Since the limitation of memory space affects the system performance, we evaluated all the three cases discussed in Section 3; *Case 1:* unlimited memory space for both proxies and peers, *Case 2:* unlimited memory space for proxies, and *Case 3:* limited memory space for both proxies and peers. We also examined the performance of the two cases for peers' mobility described in Section 3; limited and unlimited.

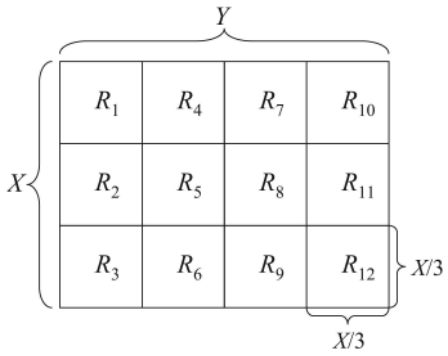

Fig. 2. Simulation area.

6.1 Simulation Model

In our simulation experiments, we assume a situation where members engaged in a collaborative work such as rescue operations share information for efficiency of their own task. 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 seconds 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 $X/3 \times X/3$, $R = \{R_1, \ldots, R_{12}\}$ (see Fig. 2). Here, ratio $X : Y$ is kept to 3:4. In our experiments, X is changed as a variable parameter in the range from 300 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 ($M = M_1, \ldots, M_{240}$). M_i ($i = 1, \ldots, 12$) is the proxy of region R_i, and M_j ($j = 13, \ldots, 240$) is a peer that initially exists in region $R_{(m \bmod 12)}$ if (m mod 12) is not 0 or R_{12} if (m mod 12) is 0. The number of data items in the entire network is 500 ($D = D_1, \ldots, D_{500}$). For the purpose of simplicity, we assume that all data items are of the same size.

We assume two cases for peers' mobility; limited and unlimited. In the limited mobility model, each peer does not go beyond its initially assigned region; thus, every region contains 20 static peers. Each peer moves according to the random waypoint model [3], where each host selects a random destination in its assigned region. In the unlimited mobility model, each peer moves according to the random waypoint model, where each host selects a random destination in the whole area. In both models, the pause time and the maximum movement speed are set as 0 second and 2 m/s, respectively, assuming that peers move in a walking speed.

The communication range of each mobile host is a circle with a radius of 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. While the GC and LC protocols adopt 2PC in phase 3, the possible approaches for this aim depend on the implementation; thus, we just count the hopcount for one round of message transmissions.

We assume three different cases for memory space. The first one is the unlimited case in which every proxy and peer replicates all the 500 data items. The second one is the unlimited case with proxies in which proxies replicate all the 500 data items and peers do not replicate any data items. The last one is the limited case in which each mobile host replicates 150 data items by using SRA and ISRA, where the minimum number of replicas in each region for every data item is set as 1. The last case is evaluated only in the limited mobility model because we do not want to consider cases where there is no replica of a certain data item in some region, which happen when all the replica holders go out from the region.

Read and write frequencies of each peer to data items are 0.02/second and 0.002/second, respectively. The read/write probability, q_j, to D_j in the entire network follows the Zipf distribution [23] and is expressed by the following equation:

$$q_j = \frac{j^{-0.6}}{\sum_{m=1}^{500} m^{-0.6}}. \quad (1)$$

Equation (1) shows that data items with smaller identifiers are requested more frequently. This is for simplicity of discussion, but we can arbitrarily change the order without losing generality. In this read/write model, every proxy and peer has the same access characteristics. This is to properly examine the impact of the replication strategy.

In TC, the validity period for read operations is set to 10 seconds. In GC and LC, $|QLR_{ij}|$ $(i = 1, \ldots, 12, j = 1, \ldots, 500)$ is set to $\lfloor P_{ij}/2 \rfloor$ and $|QLW_{ij}|$ is set to $P_{ij} - |QLW_{ij}|+1$. In GC, $|QR|$ is set to 6 (= 12/2) and $|QW|$ is set to 7. In LC, every data operation request is processed through the proxy. In TC, a write operation is not propagated to other peers and proxies.

In the simulations, we examined the success ratios of read and write operations, the message traffic, and data traffic to process a read/write operation during 200,000 units of simulation time (1,000,000 seconds). The success ratio is defined as the ratio of successful read/write operations to all requests of read/write operations issued during the simulation time. The message traffic is defined as the average of the total hopcount for message exchanges to process a read/write operation excluding transmissions of data items. The data traffic is defined as the average of the total hopcount to transmit a data item to perform a (successful) read/write operation. Here, because the sizes of data items change depending on kinds of data and also on applications, we do not assume any particular sizes for both messages and data items. The actual traffics caused by message and data transmissions can be calculated by multiplying the message and data traffics obtained in our experiments by their sizes.

We assume that every protocol execution finishes within a unit of simulation time, i.e., 5 seconds. This is usually true in a real environment, because even in GC, one round of message exchanges in the entire network takes less than 1 or 2 seconds, and thus, the entire procedure that requires three rounds of messages or data exchanges takes less than 5 seconds. Even for transmission of a data item, it takes almost the same time if the data size is not very large. In MANETs, data items shared among mobile hosts are basically not very large, e.g., from a few dozen bytes to a few Megabytes.

To examine the reliability of the simulation results, we calculated breadths of 90 percent confidence intervals for the simulation results using the *Batch Means method* [6], where the batch size is 20,000 units of time and the number of batches is 10. The results show that the simulations were sufficiently converged, where the breadths of the confidence intervals were a few percent in most cases and at most 10 percent of the average values.

In the experiments, we only evaluated the four primitive consistency levels, GC, LC, PC, and TC, because AC is a hybrid version of the primitive ones. We basically assumed that neither network topology change nor peer failure occurs during the protocol execution in every consistency level. This is because the probability that the network topology changes and a peer failure occurs during the protocol execution is generally small. However, we also provide some simulation results where we assumed both the network topology change and peer failure during the protocol execution.

6.2 Case 1-1: Unlimited Memory and Limited Movement

First, assuming the case where the memory space of all proxies and peers is unlimited and each of them moves around inside a particular region, we examine the performance of the protocols to achieve the four primitive consistency levels.

Fig. 3 shows the simulation results. In all graphs, the horizontal axis indicates area size, X. The vertical axis indicates success ratio in the cases of Figs. 3a and 3b, message traffic in the cases of Figs. 3c and 3d, and data traffic in the cases of Figs. 3e and 3f. From Figs. 3a and 3b, the success ratios of both read and write operations in GC and LC become lower as the area size becomes larger. This is because the connectivity among mobile hosts becomes lower; thus, the necessary number of locks on replicas cannot be set with high probability. The differences in success ratio between write and read operations are not big in both GC and LC because the difference in quorum sizes between write and read operations are at most 1. We can see an interesting fact that when the area size is larger than 450, the success ratio in GC suddenly becomes lower, but in LC, it remains high. This fact shows that even when the connectivity among mobile hosts is still high in each region, the connectivity among proxies becomes low. It is expected that the employed mobility model (random waypoint model) accelerates this characteristic because in this model peers tend to locate near the center of the region.

The success ratio of write operations in TC and those of write and read operations in PC are always 1 because every peer replicates all data items and operations can be executed locally. The success ratio of read operations in TC becomes lower as the area size becomes larger. This is because when the connectivity is low, mobile hosts cannot access valid replicas held by connected mobile hosts with high probability. Comparing LC and TC, TC always gives a

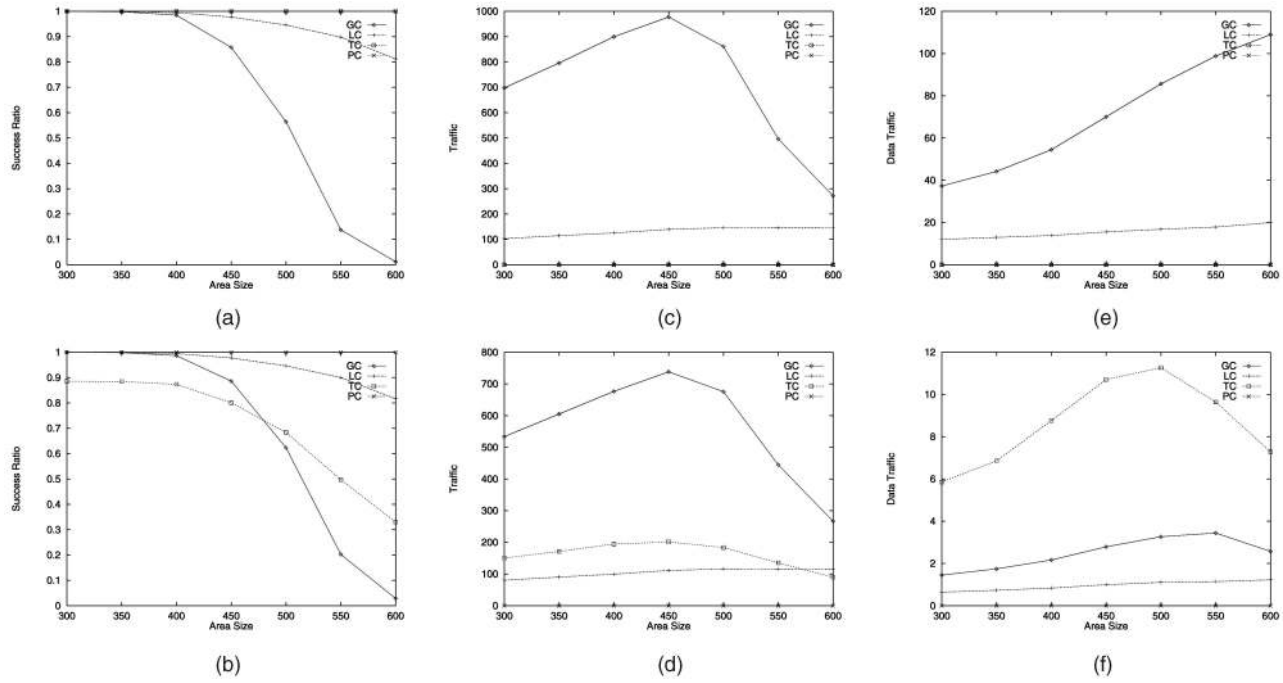

Fig. 3. Effects of the area size (Case 1-1: unlimited memory and limited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

lower success ratio, although TC weakens consistency from the temporal perspective. This is because in this experiment a peer that performed a write operation does not propagate the operation to other peers, thus replicas held by peers are only refreshed by themselves, which results in many replicas expiring the validity period.

From Figs. 3c and 3d, the message traffic of write and read operations in GC and that of write operations in TC first become higher and then lower from a certain point $(X = 450)$ as the area size becomes larger. The reason why the message traffic first becomes higher in GC is that the proxy that received a request of a write/read operation from a peer fails more times to find proxies that can set the necessary number of local locks in their responsible regions. The message traffic of write operations in TC first becomes higher because the request-issuing peer fails more times to find a proxy or peer that holds a valid replica. The message traffic in both GC and TC (write operations) then becomes lower because the number of proxies and peers that a request-issuing peer can communicate with becomes lower. This fact can be confirmed from the results in Figs. 3a and 3b in which the success ratios in these cases become lower. Of the four protocols, GC produces the highest message traffic, and LC produces much lower than GC and TC (for write operations).

The message traffic of write and read operations in LC is barely affected by the area size. This is because a request-issuing peer always multicasts the request to all the replica holders in the region. The reason why the message traffic in LC slightly increases as the area size becomes larger is that hopcounts between two mobile hosts become higher. Since the connectivity among peers in the same region is still high even when the area size is large, the message traffic does not decrease in LC. Here, the differences in message traffic between write and read operations in GC and LC are mainly due to phase 3 in which read operations require one-way message transmission. Obviously, the message traffic of write operations in TC and those of write and read operations in PC are always 0.

From Figs. 3e and 3f, the data traffic is much lower than the message traffic for both write and read operations. However, the values do not represent actual traffic because the data size is not considered. If the sizes of messages and data items are given, it is determined which one is dominant. Comparing write and read operations, the data traffic for write operations is much higher than that for a read operation. This is obvious because a read operation just requires a unicast from the closest peer having the latest replica to the proxy or the request-issuing peer, while a write operation requires a multicast of the data item of the new version.

In GC and LC, the data traffic for both write and read operations basically keeps increasing as the area size becomes higher. This is because hopcounts between two peers tend to be larger and the data traffic is measured only when a data operation request succeeds, i.e., fail cases are ignored. The data traffic for read operations in TC is much higher than other protocols, because there are fewer valid replicas in TC as mentioned above, and request-issuing peers have to obtain valid replicas from faraway peers.

Here, we examine the impact of differences in peer density among regions. Of course, we can estimate the performance of LC in each region from the results in Fig. 3 even when the numbers of peers differ among regions. However, the difference in peer density affects the performance of GC and TC because existence of sparse regions between dense regions may cause network partitioning. To examine this, we changed the simulation settings to achieve an environment where there are two kinds of regions, *dense* and *sparse*. Specifically, six regions having odd

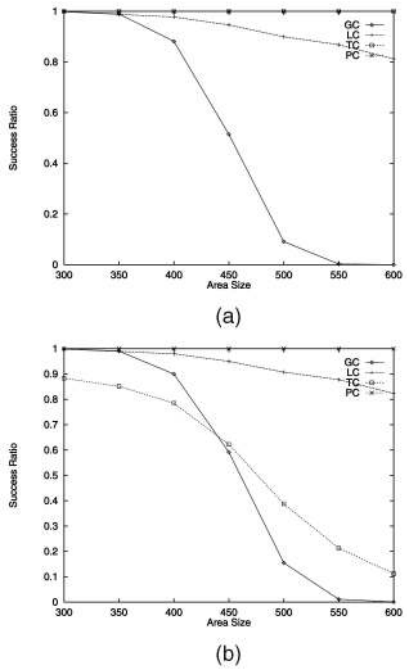

Fig. 4. Case 1-1: unlimited memory and limited movement (different density). (a) Success ratio (write). (b) Success ratio (read).

numbered identifiers, $R_1, R_3, \ldots, R_{11}$, are assigned 10 peers (sparse regions), while the other six regions, $R_2, \ldots, R_{12}$, are assigned 30 peers (dense regions). Note that the density of peers in the entire area is the same as that in the simulation in Fig. 3. Fig. 4 shows the simulation result. Due to the limitation of space, only the success ratios are shown here. From this result, the success ratios for read and write operations in GC and that for write operations in TC drop faster than the result in Fig. 3. On the contrary, the success ratios in LC are barely affected by the difference in peer density. This result confirms that the existence of sparse regions makes the connectivity among regions worse and the success ratios for data operations decrease.

6.3 Case 2-1: Unlimited Memory with Proxies and Limited Movement

Next, assuming the case where proxies have unlimited memory space while peers have no memory and every proxy and peer moves inside a particular region, we examine the performance of the four protocols. Fig. 5 shows the simulation results. In this graph, the performance of PC is not shown because PC is defined only where every peer accesses its own replicas. From Fig. 5, most of the characteristics in Fig. 3 are preserved, but there are several remarkable differences. The most remarkable difference is that the message traffic is much reduced in Fig. 5, while the success ratios of write and read operations in GC and LC and that of read operations in TC are almost the same between Figs. 3 and 5. The reduction of message traffic is not surprising because a proxy that received a request from the request-issuing peer or the coordinator in GC does not need to send any messages to peers in the region in GC and LC. Also in TC, since a request-issuing peer can directly unicast a request to proxies, the message traffic is reduced.

The second remarkable difference is the results for write operations in TC. In Fig. 5, the success ratio of write operations in TC is not always 1 but becomes lower as the area size becomes larger, and the message traffic is not 0. This is because a request-issuing peer has to perform a write operation on a replica held by a proxy. Here, the performance in TC is almost the same as that in LC in which a request succeeds when a request-issuing peer connects to the proxy in its region.

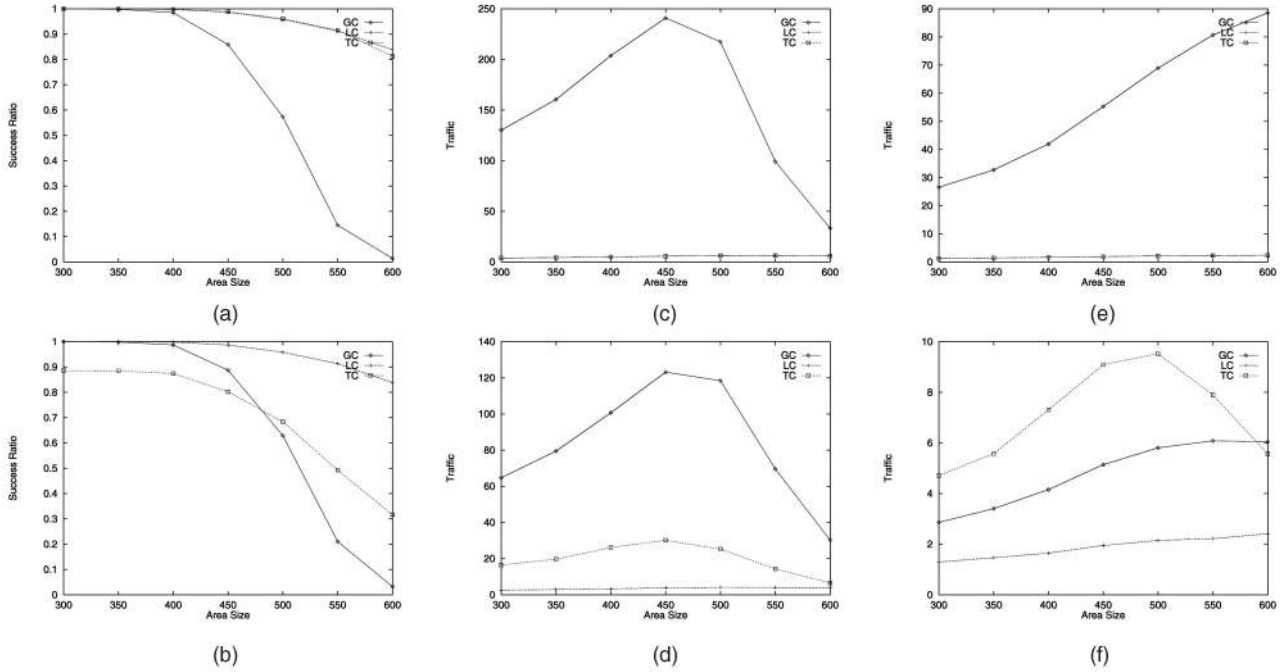

Fig. 5. Effects of the area size (Case 2-1: unlimited memory with proxies and limited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

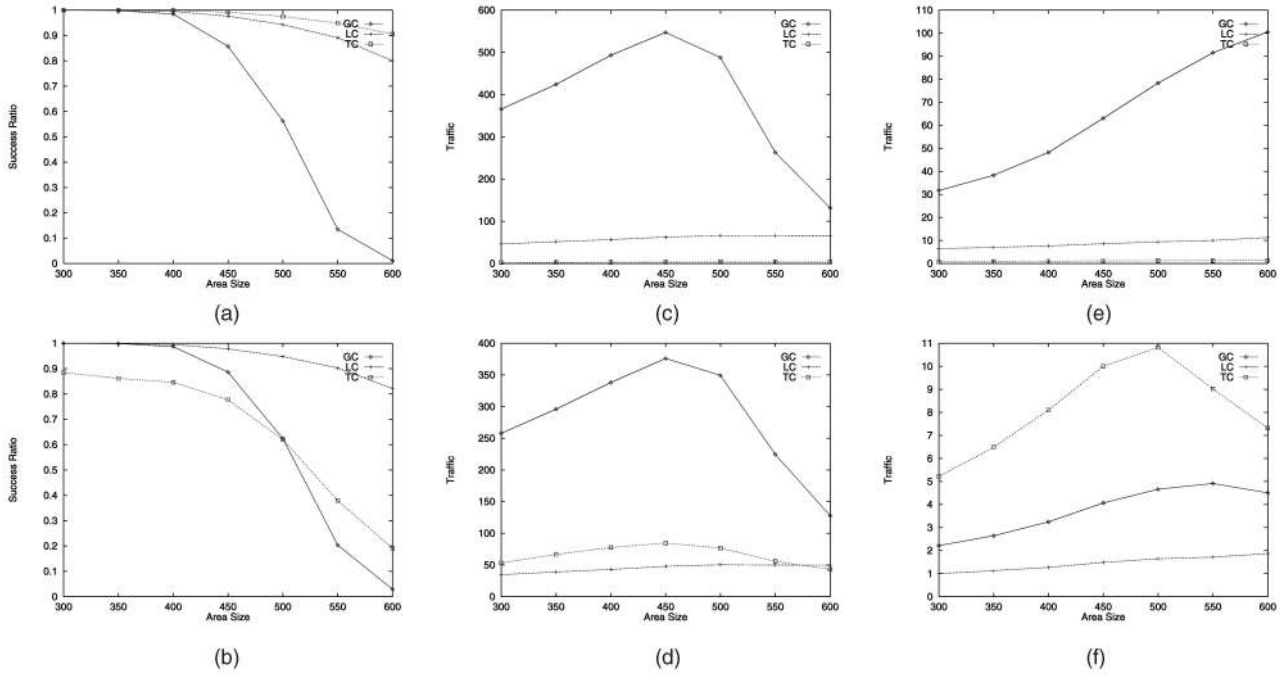

Fig. 6. Effects of the area size (Case 3-1: limited memory (SRA) and limited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

The last remarkable difference is data traffic in LC and GC. While the data traffic for write operations is reduced from the result in Fig. 3e, that for a read operation is much increased. This is because in the case of Fig. 3, it often happens that the request-issuing peer holds the latest replica.

From these results, we can confirm the following facts. When all applications running in the network require GC or LC, it is better in most cases to restrict replication so that only proxies replicate all data items and other peers hold no replicas. However, when data items are relatively large compared with the message size and read operations are issued much more frequently than the write operations, aggressive replication that creates more replicas should be better.

6.4 Case 3-1: Limited Memory and Limited Movement

We examine the performance of the proposed protocols assuming the cases where proxies and peers have limited memory space for replication (SRA and ISRA), and every peer moves inside its region. Figs. 6 and 7 show the simulation results. For the same reason as in Fig. 5, the performance of PC is not shown here. From these results, the characteristics are almost the same between the two cases, SRA and ISRA; however, several interesting facts can be seen.

Similar to the result in Fig. 5, the number of replicas held by peers barely affects the success ratios in GC and LC. As for TC, by comparing the two cases, SRA and ISRA, having more replicas of more accessed data items, i.e., SRA, slightly increases the success ratio of write operations.

The message traffic in GC and LC differs among the two cases, where having fewer replicas of more accessed data items (ISRA) reduces the message traffic. As for the data traffic in GC and LC, having less replicas of more accessed data items slightly reduces the data traffic for write operations and slightly increases for read operations. Together with the results in Fig. 5, these results on the performance of the quorum-based approaches are completely different from a general knowledge that having more replicas of more accessed data items reduces the message traffic as well as increases the data availability, i.e., success ratios of read operations.

As for TC, the message traffic of write operations is reduced by having more replicas of more accessed data items (SRA). This is due to the same reason as that in Fig. 5.

We conducted some other experiments in different settings of peers' memory sizes for replication. We also conducted some experiments in which different peers have different memory sizes. All the results of these experiments basically showed the same characteristics as the results in Figs. 6 and 7. Due to the limitation of space, we omit the details.

6.5 Case 1-2: Unlimited Memory and Unlimited Movement

Next, we examine the other movement pattern, where peers move across regions. Fig. 8 shows the simulation results in the case that proxies and peers have unlimited memory space and replicate all data items. From the results, we can observe several interesting facts. First, unlimited movement of peers degrades the success ratios of both read and write operations in LC but improves those in GC. As for GC, points where the success ratios start to drop down are later and angles of dip are more modest than the results in Fig. 3. On the contrary, LC shows the reverse characteristics. This is due to the characteristic of the random waypoint model, where more nodes tend to locate near the center of the whole area. Thus, the connectivity among peers in different regions that exist near the center of the whole area becomes

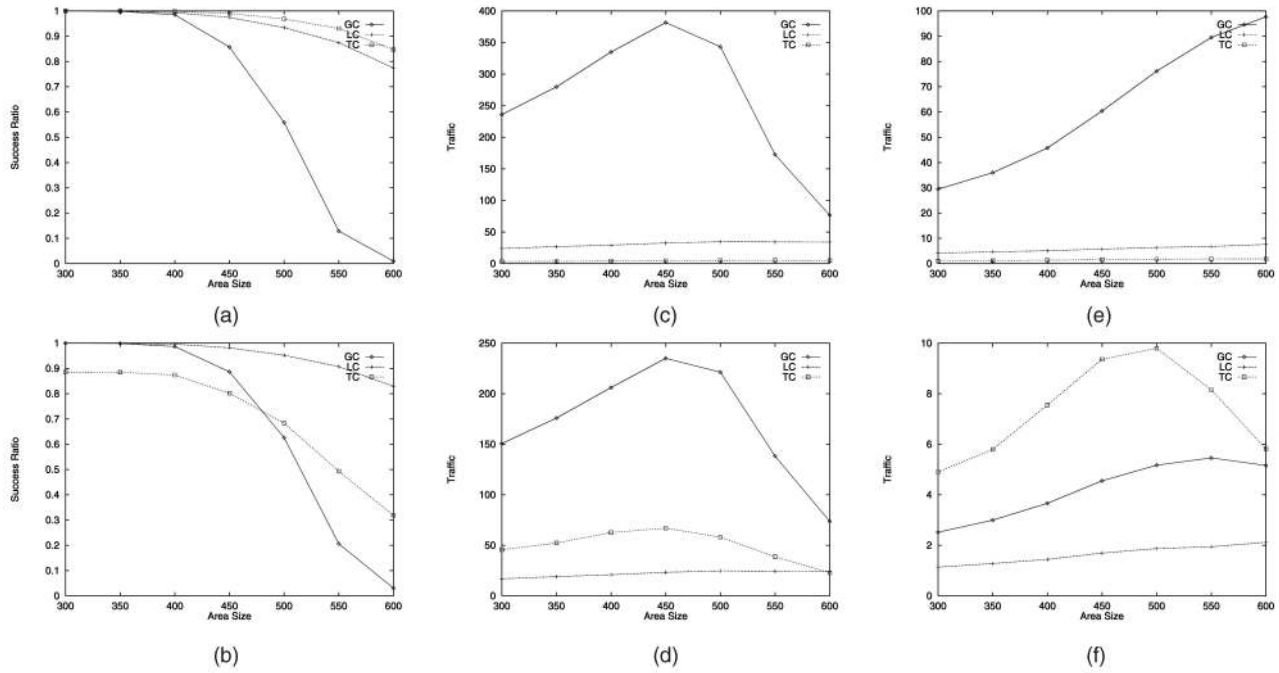

Fig. 7. Effects of the area size (Case 3-1: limited memory (ISRA) and limited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

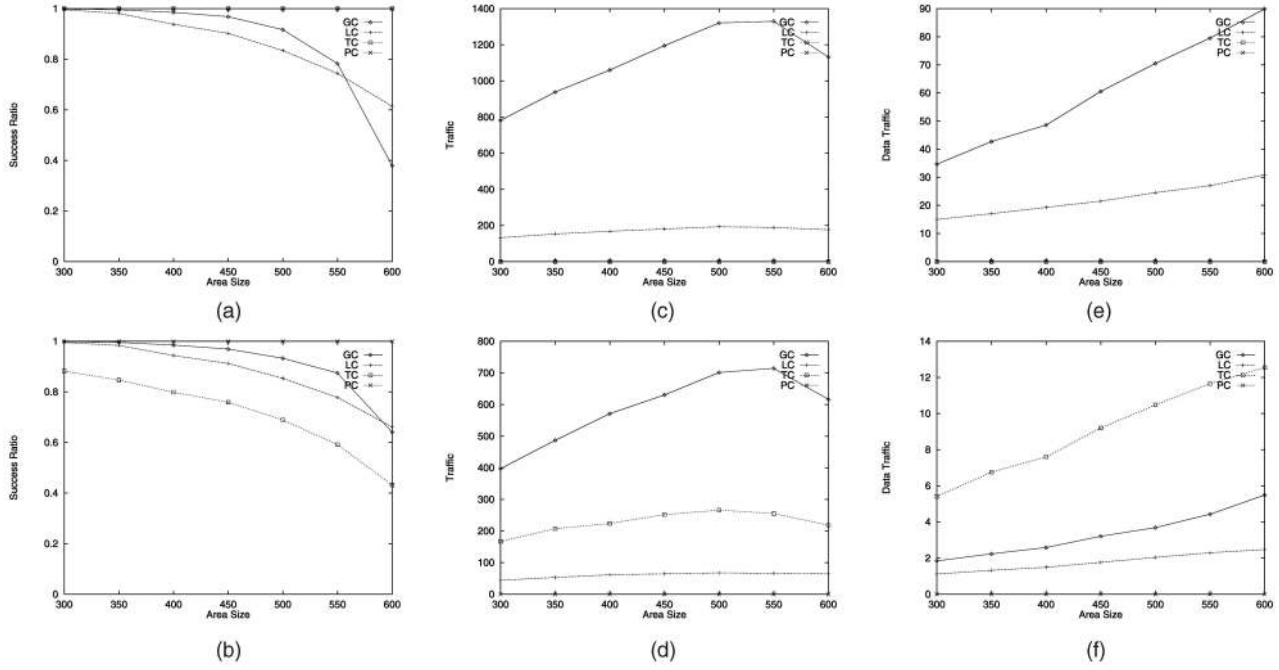

Fig. 8. Effects of the area size (Case 1-2: unlimited memory and unlimited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

higher and the success ratio in GC increases. The success ratio of read operations in TC also increases due to the same reason. On the other hand, since the density of peers in regions that exist far from the center tends to be lower, the success ratios of write and read operations in these regions decrease, which results in a decrease of success ratio in LC in the entire network. Here, in GC, this does not much affect the success ratio because the necessary number of global locks can be set on proxies in regions near the center of the area.

To verify the above discussions, we examine the average number of peers and success ratios of operations in each region, focusing on the case of $X = 450\,\mathrm{m}$. As for the average number of peers in each region, we show both peers that successfully register on entrance into each region and peers that geographically (actually) exist in each region independent of whether or not the peers succeed to register on entrance. Fig. 9 shows the experimental result. In this figure, the horizontal axis indicates the region number shown in Fig. 2 and the vertical axes indicate the number of

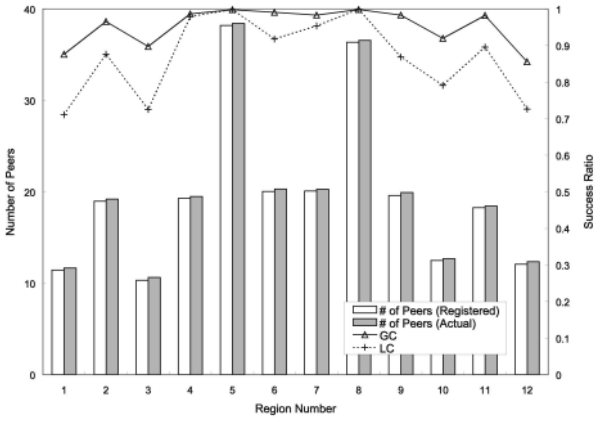

Fig. 9. Average number of peers and success ratio in each region.

peers in each region for bar graphs and the success ratios of write operations for line graphs. From the result, it can be seen that the difference between the number of registered peers and that of geographically existing peers is very small, which shows that peers mostly succeed to register on their entrance to new regions. Also, we can confirm the facts discussed in the previous paragraph from the result on success ratios in each region.

From Figs. 8c and 8d, we can observe an interesting fact in the result for message traffic in GC and LC. Compared with the result in Fig. 3, the differences in message traffic between write and read operations become much larger. This is because in an environment where peers move across regions, the local quorum size for a read operation on a data item is decreased as peers having an old replica of the data item exit from the region. This makes the local quorum size for a read operation smaller, and thus, the message traffic becomes lower. Surprisingly, this is helpful to improve the data success ratio of read operations in GC, as shown in Fig. 8b.

On the other hand, the message traffic for write operations in GC is higher than that in Fig. 3. This is due to the increase in numbers of peers in regions that exist near the center of the whole area and the increase in connectivity among these peers in different regions.

6.6 Impact of Mobility Model

As mentioned above, the results of our simulations are affected by the employed mobility model (random waypoint model). In this section, to examine the impact of the mobility model, we measure the performance of the proposed protocols assuming another well-known mobility model, called the random walk model. In this model, each mobile node randomly determines the movement direction from all directions and randomly determines the movement speed from 0 to 2 m/s at every unit of simulation time. In the random walk model, peers distribute randomly (almost uniformly) in each region. Because the performance of our proposed protocols depends on the geographical distribution of peers, we think examining the random walk model is enough to examine the side effects of different mobility models.

Figs. 10 and 11 show the results of the simulations in the two cases where peers move inside regions (limited movement) and move across regions (unlimited movement),

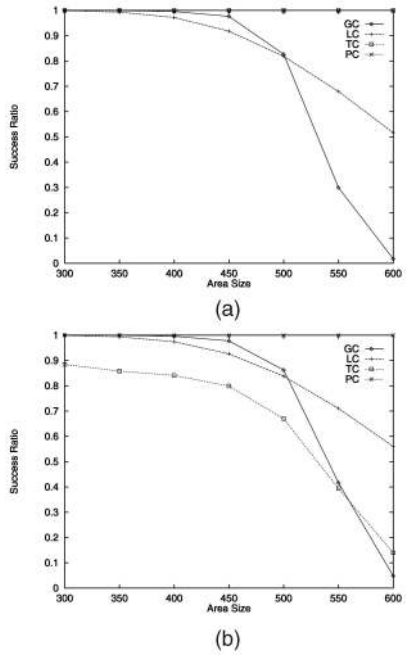

Fig. 10. Case 1-1: unlimited memory and limited movement (random walk). (a) Success ratio (write). (b) Success ratio (read).

respectively. In both cases, we assume that peers have unlimited memory space for replication. Due to the limitation of space, only the success ratios are shown here. From these results, while the general features are basically reserved even when using the random walk model, we can confirm the side effects of the mobility models. Specifically, since peers tend to distribute almost uniformly in each region when using the random walk model, the success ratios in LC become lower than the case using the random

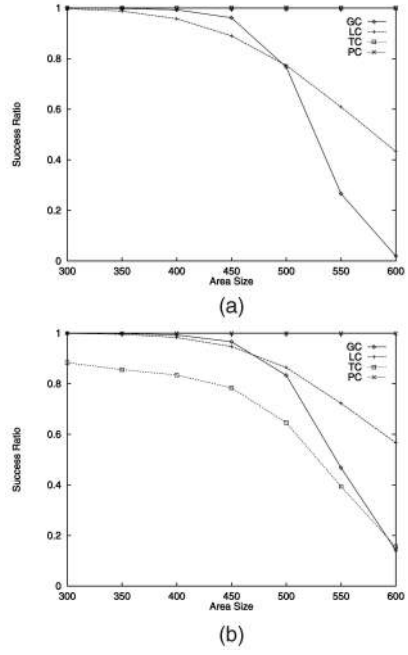

Fig. 11. Case 1-2: unlimited memory and unlimited movement (random walk). (a) Success ratio (write). (b) Success ratio (read).

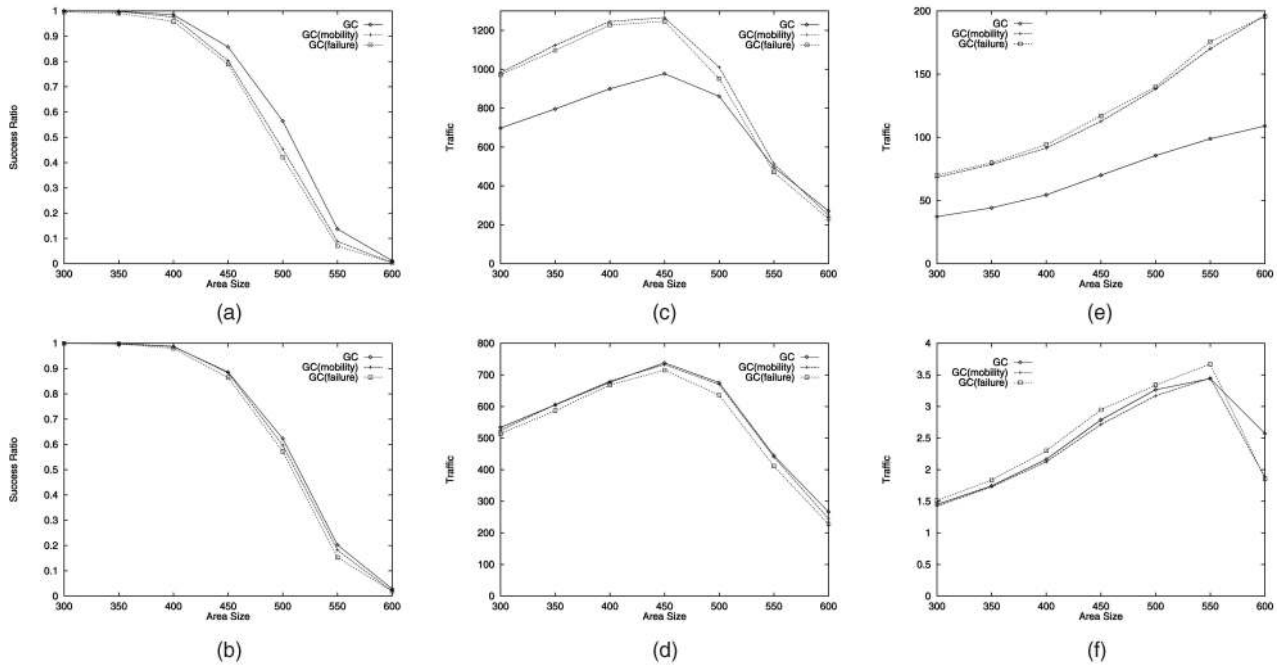

Fig. 12. Impact of node failure (Case 1-1: unlimited memory and limited movement). (a) Success ratio (write). (b) Success ratio (read). (c) Message traffic (write). (d) Message traffic (read). (e) Data traffic (write). (f) Data traffic (read).

waypoint model for both limited and unlimited movements. As for GC, the success ratios become higher and the performance drop points become later than the case using the random waypoint model for limited movement, while those for unlimited movement show opposite features. Comparing the limited and unlimited movement cases, in every protocol, we cannot find big differences in success ratios between the two cases. This is because peers tend to distribute almost uniformly in the entire area in both cases.

6.7 Impact of Node Failure and Topology Change

In the above simulations, we assume that neither topology change nor node failure occurs during the protocol execution. In this section, we examine the impact of topology change and node failure. Since the execution time of the three protocols to achieve LC, TC, and PC is very short, e.g., less than 1 or 2 seconds, the impact of topology change and node failure during the protocol execution is generally small enough to be negligible. Therefore, we focus only on the protocol of GC, where its execution time is usually a few seconds to 5 seconds. Here, since the difference in time scale between our simulations (5 seconds) and message transmissions in a real environment (millisecond) is very large, it is difficult to fairly examine the impact of topology change and node failure.

Thus, we assume a very simple model to this end in which the protocol execution of GC takes 5 seconds (a unit of simulation time). Then, if the coordinator of a read/write operation connected with enough number of proxies and peers to construct global and local quorums at the request issued time, it is checked whether these connections are still available at the next simulation step, i.e., 5 seconds later. If some of the connections are not available, it represents that a failure happened during the protocol execution due to either topology change or node failure. In that case, we simply assume that there are two cases in which the failure occurred during phase 2 or during phase 3, and these cases occurred with the same possibility. As for peer failure, we assume an environment where nodes fail with realistic frequencies, i.e., periodic 5-minute failures after running 3 hours for changing or recharging a battery.

Fig. 12 shows the simulation results in the case that proxies and peers have unlimited memory space and peers move inside regions (random waypoint). In this experiment, we show the performance of the two cases; the case considering the topology change during the protocol execution (denoted by "GC(mobility)") and the case considering both the topology change and peer failure (denoted by "GC(failure)"). For comparison, we also show the performance of the case where we assume neither topology change nor peer failure during the protocol execution (denoted by "GC"), which is identical to that in Fig. 3.

Overall, it can be seen that the impact of topology change is dominant, and that of node failure is basically small. Moreover, we can observe several remarkable facts. First, while the impact of topology change and peer failure on the success ratio is not very large, that for write operations is larger than that for read operations. This is because it is much easier to find an alternative peer that holds the latest replica for read operations than to collect the necessary number of peers to reconstruct a local write quorum. Moreover, the impact becomes larger as the area size becomes higher, i.e., node density becomes lower.

Second, the impact of topology change and peer failure on message and data traffic for write operations is much larger than that for read operations. The reasons are as follows: As for message traffic, a write operation sometimes requires another message multicast to collect alternative

replica holders, while a read operation just needs to find another latest replica holder, and the information for this aim is available at phase 1 in most cases. As for data traffic, a write operation causes a large number of useless data transmissions if phase 3 fails.

We have also conducted some experiments in other conditions on memory space, i.e., Case 2-1 and Case 3-1. Due to the limitation of space, we do not present the details, but these results show almost the same characteristics.

6.8 Summary of the Simulation Results

In summary, through the simulation experiments, we found several significant and interesting facts as listed below.

When mobility of peers is limited inside their region, i.e., mobility has strong locality, LC gives higher success ratios (data availability) of both write and read operations than GC if the network is relatively sparse. This feature is more conspicuous in the case of using the random waypoint model, in which nodes tend to locate near the center of the region than the case of using the random walk model. Thus, if applications do not require the strict consistency of data operations in the entire network, we should choose LC in terms of both success ratio and traffic.

When peers move across regions by selecting the destinations randomly (i.e., random waypoint model), GC gives higher success ratios than the case of limited mobility. In a highly dynamic environment, it is not necessarily a good choice to use LC, where it can happen that LC gives a lower success ratio than GC. On the other hand, when peers move across regions based on the random walk model, the performance of each protocol is almost the same as when peers move inside particular regions. Thus, we can confirm that the performance of our protocols heavily depends on the mobility pattern, i.e., distributions of peers in the regions and entire area. Moreover, our protocol to achieve GC could appropriately change the size of the local read quorum according to the change of members in the region, which works well even in an unlimited mobility environment.

In GC and LC, due to the characteristic of the quorum system, having more replicas does not result in higher success ratios, while it produces higher message traffic for both write and read operations and higher data traffic for write operations. Thus, in most cases, we should restrict replication if the proxy has enough memory space to replicate all or most of the data items. However, having more replicas results in lower data traffic for read operations. Thus, when the data items are relatively large compared with the message size and read operations are issued much more frequently than the write operations, having more replicas should be better.

While TC weakens consistency from the temporal perspective, it gives lower success ratios and higher message traffic for read operations than LC in most cases. More specifically, performing write operations on only local replicas improves traffic for write operations but much degrades the performance for read operations. This fact shows that a weaker consistency level does not simply give better performance. Thus, we should carefully design a protocol to achieve each consistency level.

Although topology change and peer failure during the protocol execution do not much affect the success ratios in GC, it increases both message and data traffic for write operations. Therefore, in a highly dynamic environment, we have to be careful to use GC if the network resource and battery of peers are highly restricted.

7 Discussions

In this section, we briefly describe some issues open for investigation in our future work.

7.1 More Efficient Quorum Construction

In a quorum system, every pair of read and write quorums must have an intersection. To meet this, our GC protocol adopts a simple approach so that the total number of proxies contained in global write and read quorums has to be set at more than the total number of regions. However, this approach causes large communication overhead. Therefore, we will further investigate more efficient methods to construct quorums.

7.2 Replication Strategy Suitable for a Quorum System

In this paper, we examined the impact of the two typical replication strategies; SRA and ISRA. However, there are a large number of replication strategies proposed for MANETs. We will further examine the impact of these existing replication strategies and which one is optimal for each of the consistency management protocols.

7.3 Other Consistency Levels

Although our proposed consistency levels cover many applications in MANETs, we realize that they do not cover all. As part of our future work, we will consider other consistency levels suitable for MANET applications. For example, while LC requires consistency of data operations in the same region, it does not care about divergence of the values in different regions. However, some applications might require a certain degree of divergence of values among copies such as d-consistent proposed in [19]. Such a consistency level can be considered as in-between GC and LC, which requires a weaker consistency among regions (proxies). We expect that such a consistency level can be achieved by applying some reconciliation mechanisms proposed for traditional mobile databases.

7.4 Complex Transactions

In our work, we assume a simple transaction model in which every transaction will consist of a data operation (read or write). However, in a real environment, there are also many applications in which a transaction consists of multiple operations. In such an environment, consistency management based on a pessimistic policy becomes much more complex. We plan to address this issue in our future work.

8 Conclusion

Since in MANETs peers' disconnection causes frequent network partitioning, it is difficult and, in some cases, not desirable to provide traditional strong consistency of data operations. Moreover, since there are many kinds of applications possible in MANETs, there cannot be one universal optimal strategy for consistency management. Thus, we have classified consistency levels according to

applications' demand and, then, have designed protocols to achieve them.

We have conducted extensive simulations to investigate the behaviors and features of our proposed protocols. From these results, it is shown that the performance of the proposed protocols much differs with each other and that we should choose LC rather than GC in terms of both success ratio and traffic if applications do not require the strict consistency in the entire network. It is also shown that limitations of mobility and memory space have impacts on the performance. The results tell us that in most cases, we should restrict replication at peers if the proxy has enough memory space to replicate all or most of the data items.

As part of future work, we plan to consider replication strategies suitable for each consistency level. We also plan to extend our protocols to deal with more complex transactions.

ACKNOWLEDGMENTS

This research was supported in part by the International Communications Foundation and in part by the Ministry of Education, Culture, Sports, Science, and Technology, Japan through a Grant-in-Aid for Scientific Research (A) (17200006) and for Scientific Research on Priority Areas (18049050). The author would like to express his sincere appreciation to Prof. Shojiro Nishio of Osaka University for invaluable comments on this work.

REFERENCES

[1] R. Alonso, D. Barbara, and H. Garcia-Molina, "Data Caching Issues in an Information Retrieval System," *ACM Trans. Database Systems,* vol. 15, no. 3, pp. 359-384, 1990.

[2] D. Barbara and H. Garcia-Molina, "The Reliability of Vote Mechanisms," *IEEE Trans. Computers,* vol. 36, no. 10, pp. 1197-1208, Oct. 1987.

[3] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proc. ACM MobiCom,* pp. 159-164, 1998.

[4] G. Cao, L. Yin, and C.R. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks," *Computer,* vol. 37, no. 2, pp. 32-39, 2004.

[5] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM,* pp. 177-190, 2002.

[6] R.W. Conway, "Some Tactical Problems in Digital Simulation," *Management Science,* vol. 10, pp. 47-61, 1963.

[7] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization Using Reference Broadcasts," *Proc. Fifth Symp. Operating System Design and Implementation (OSDI '02),* pp. 147-163, 2002.

[8] L.D. Fife and L. Gruenwald, "Research Issues for Data Communication in Mobile Ad-Hoc Network Database Systems," *SIGMOD Record,* vol. 32, no. 2, pp. 42-47, 2003.

[9] T. Hara, "Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility," *Proc. IEEE INFOCOM,* pp. 1568-1576, 2001.

[10] T. Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update," *ACM-Kluwer J. Mobile Networks and Applications,* vol. 8, no. 4, pp. 343-354, 2003.

[11] T. Hara and S.K. Madria, "Consistency Management among Replicas in Peer-to-Peer Mobile Ad Hoc Networks," *Proc. IEEE Symp. Reliable Distributed Systems (SRDS '05),* pp. 3-12, 2005.

[12] T. Hara and S.K. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 5, no. 11, pp. 1515-1532, Nov. 2006.

[13] Y. Huang, P. Sistla, and O. Wolfson, "Data Replication for Mobile Computer," *Proc. ACM SIGMOD '94,* pp. 13-24, 1994.

[14] D.B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proc. IEEE Workshop Mobile Computing Systems and Applications (WMCSA '94),* pp. 158-163, 1994.

[15] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information Dissemination in Partitionable Mobile Ad Hoc Networks," *Proc. IEEE Symp. Reliable Distributed Systems (SRDS '99),* pp. 4-13, 1999.

[16] J. Luo, J.P. Hubaux, and P. Eugster, "PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems," *Proc. ACM MobiHoc,* pp. 1-12, 2003.

[17] S.K. Madria, "Timestamps to Detect R-W Conflicts in Mobile Computing," *Proc. Int'l Workshop Mobile Data Access (MDA '98),* pp. 242-253, Nov. 1998.

[18] D. Malkhi, M.K. Reiter, and A. Wool, "Probabilistic Quorum Systems," *Information and Computation,* vol. 170, no. 2, pp. 184-206, 2001.

[19] E. Pitoura and B. Bhargava, "Data Consistency in Intermittently Connected Distributed Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 11, no. 6, pp. 896-915, Nov./Dec. 1999.

[20] E. Pitoura, P.K. Chrysanthis, and K. Ramamritham, "Characterizing the Temporal and Semantic Coherency of Broadcast-Based Data Dissemination," *Proc. Int'l Conf. Database Theory (ICDT '03),* pp. 410-424, 2003.

[21] K. Ramamritham and C. Pu, "A Formal Characterization of Epsilon Serializability," *IEEE Trans. Knowledge and Data Eng.,* vol. 7, no. 6, pp. 997-1007, Dec. 1995.

[22] K. Rothermel, C. Becker, and J. Hahner, "Consistent Update Diffusion in Mobile Ad Hoc Networks," Technical Report 2002/04, Computer Science Dept., Univ. of Stuttgart, 2002.

[23] G.K. Zipf, *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.

Takahiro Hara received the BE, ME, and DrE degrees in information systems engineering from Osaka University, Japan, in 1995, 1997, and 2000, respectively. He is currently an associate professor in the Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University. He served as a program chair of the IEEE International Conference on Mobile Data Management (MDM '06). He is currently serving as a program chair of the IEEE International Conference on Advanced Information Networking and Applications (AINA '09). He served and is serving on various international conferences such as the IEEE International Symposium on Network Computing and Applications (NCA), the International Conference on Database Systems for Advanced Applications (DASFAA), ACM MobiHoc, and the Annual ACM Symposium on Applied Computing (SAC). His research interests include distributed databases, peer-to-peer systems, mobile networks, and mobile computing systems. He is a senior member of the IEEE, a member of the ACM, and a member of three other learned societies.

Sanjay Kumar Madria received the PhD degree in computer science from Indian Institute of Technology, Delhi, India, in 1995. He is an associate professor in the Department of Computer Science, Missouri University of Science and Technology, Rolla. Earlier, he was a visiting assistant professor in the Department of Computer Science, Purdue University, West Lafayette, Indiana. He has published more than 125 journal and conference papers in the areas of mobile computing, sensor networks, security, and XML data management. He is a recipient of a UMR Faculty Excellence Award, an AFRL summer faculty fellowship, and a JSPS visiting fellowship among others. His research is supported by multiple grants from the US National Science Foundation, US Department of Energy, the UM research board, AFRL, and from industry. He is a senior member of the IEEE and a distinguished speaker of the IEEE and the ACM.