12 Jul 2010

Investigating Impact of Quorum Construction on Data Processing in Mobile Ad Hoc Networks

Takahiro Hara

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

Shojiro Nishio

Recommended Citation

T. Hara et al., "Investigating Impact of Quorum Construction on Data Processing in Mobile Ad Hoc Networks," *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 548 - 555, article no. 5474750, Institute of Electrical and Electronics Engineers, Jul 2010.
The definitive version is available at https://doi.org/10.1109/AINA.2010.142

2010 24th IEEE International Conference on Advanced Information Networking and Applications

Investigating Impact of Quorum Construction on Data Processing in Mobile Ad Hoc Networks

Takahiro Hara
Department of Multimedia Eng.
Osaka University
Email: hara@ist.osaka-u.ac.jp

Sanjay K. Madria
Department of Computer Science
Missouri Univ. of Science and Technology
Email: madrias@mst.edu

Shojiro Nishio
Department of Multimedia Eng.
Osaka University
Email: nishio@ist.osaka-u.ac.jp

***Abstract*—In a mobile ad hoc network (MANET), since mobility of mobile hosts causes frequent network partitioning, consistency management of data operations on replicas becomes a crucial issue. In our previous work, we have defined several consistency levels for MANET applications and designed protocols to achieve these consistency levels. These protocols are mainly based on a dynamic quorum system to cope with network partitioning and node and network failures. In this paper, we further investigate the impact of quorum construction on the system performance through simulation studies. Specifically, we change the number of mobile hosts that replicate data items and which hosts replicate each data item in the simulations and examine the impact on the system performance in terms of data availability and traffic.**

***Index Terms*—mobile ad hoc networks, quorum, consistency management, data replication**

I. Introduction

Recently, there has been increasing interest in *mobile ad hoc networks (MANETs)* which are constructed only by mobile hosts with wireless communication facilities [1], [3], [6], [12]. In a MANET, movement of mobile hosts often causes disconnections of nodes and sometimes network partitioning. At the point of network partitioning, data in two separated networks become inaccessible to each other. Therefore, preventing the deterioration of data availability at the point of network partitioning is a significant issue in MANETs [8], [11], [14]. To improve data availability, data replication is the most promising solution [4], [7], [9], [19].

In [10], we assumed that the area in which mobile hosts can move around is divided into several regions and proposed several consistency management protocols for data operations on replicas based on the regions. In the GC (Global Consistency) protocol, consistency of data operations is maintained in the entire network and it can be achieved in a hierarchical manner; among mobile hosts in each region and among proxies that manage a region. In the LC (Local Consistency) protocol, consistency is maintained only in each region. Although these two protocols have different consistency strictness from the spatial perspective, both protocols are based on a *quorum system* that has been proposed for distributed databases [2]. 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.

The GC and LC protocols adopt a dynamic quorum system which dynamically constructs quorums for read and write operations rather than static ones to improve the flexibility of quorum construction. In [10], we aimed to simplify the design of these protocols, thus, we adopted a simple approach for quorum construction. Specifically, these protocols set the quorum size for a read operation, $|QW|$, and that for a read operation, $|QR|$, are determined where the condition, $|QW| + |QR| > P$, is satisfied. Here, P is the total number of mobile hosts having a replica of the target data item in a region of interest. Owing to condition $|QW| + |QR| > P$, every read operation can read the latest version in the region. However, this approach requires a large number of message transmissions for quorum construction because the quorum sizes likely become large.

In this paper, to solve this problem, we address efficient quorum construction and investigate the impact of quorum construction on the system performance through simulation studies. More specifically, we change the number of mobile hosts that replicate data items and which hosts replicate each data item in the simulations and examine the impact on the system performance in terms of data availability and traffic.

The remainder of this paper is organized as follows. In section II, we introduce some related work and our previous work. In section III, we explain the system model. We present how to determine replica holders (quorum size) that construct quorums in section IV. In section V, we show the simulation results. Finally we conclude this paper in section VI.

II. Related Work and Our Previous Work

In this section, we first present a quorum system for database consistency management which is a basis of our work. Then, we introduce several conventional approaches based on a quorum system for distributed database systems and our previous methods for MANET presented in [10].

A. Quorum Systems

Quorum-based consistency management is a typical technique for networks in which node and network failures often occur and the networks are sometimes partitioned [2]. In a quorum system for database consistency management, a read (write) operation is performed on only replicas held

1550-445X/10 $26.00 © 2010 IEEE
DOI 10.1109/AINA.2010.142

by database servers that form a read (write) quorum. Here, these read and write quorums are constructed so that every pair of read and write quorums have an intersection. When a client issues an update request for a data item, it performs a write operation on replicas held by all database servers in a write quorum so that replicas of the latest version exist in the quorum. On the other hand, when a client reads a data item, it performs the read operation on replicas held by all database servers in a read quorum. Since there is an intersection between write and read quorums, at least one database server 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.

Except for the condition that every pair of read and write quorums have an intersection, quorums can be constructed in arbitrary ways according to the system requirements. It is generally desirable that the sizes of quorums become small as much as possible.

For quorum construction, there are two typical approaches: static and dynamic. In static quorum construction, one or more static quorums for read and write operations are respectively constructed in advance where the above intersection condition is satisfied. The main advantage of this approach is that it can reduce the quorum sizes. In an extreme case, both read and write quorums can be set as a same particular node, i.e., the quorum size is 1. However, this approach also has a disadvantage that it is not flexible for node failures and network partitioning. On the contrary, in dynamic quorum construction, no static quorums are constructed but quorums are dynamically constructed according to a certain rule in order to satisfy the the intersection condition. The most simple way is setting the quorum size for a read operation $|QR|$ and that for a write operation $|QW|$ are set on condition that $|QR|+|QW|>P$, where P is the number of replica holders. Then, quorums can be arbitrary constructed where only the restriction of the quorum sizes is met.

B. *Efficient Quorum Construction*

In highly dynamic network environments, dynamic quorum systems are effective rather than static ones. Therefore, various studies have been made to construct dynamic quorums efficiently. The main objective of these conventional studies is reduction of the quorum size. A simple way to meet the above condition, $|QR| + |QW| > P$, is constructing quorums with larger than half of replica holders. However, such a majority based approach makes the quorum size still large. To solve this problem, *hierarchical quorum systems* [15] and *grid-based and triangle-based quorum systems* [5], [16] have been proposed.

In hierarchical quorum systems, replica holders are hierarchically grouped into clusters and a dynamic quorum system such as a majority-based approach is used at each level. As the number of hierarchical levels gets larger, the number of nodes involved in the quorums becomes smaller, e.g., $O(1/2^l)$ where the number of levels is l and a majority-based approach is used. However, the protocol to achieve the quorum system become more complex and takes longer time for execution.

In grid-based and triangle-based quorum systems, quorums are constructed by replica holders that respectively form grids and triangles in a certain logical plane based on node identifiers. In these quorum systems, the size of quorums becomes $O(\sqrt{n})$ where n is the number of nodes in the entire system. However, such an identifier based approach works well only in environments where the connectivity between every pair of nodes, e.g., bandwidth and latency, is almost same in the entire network or the identifier is well assigned by taking into account the geographical condition or the physical network condition. Therefore, it is generally difficult to apply these quorum systems to MANETs.

C. *Efficient Quorum Construction in MANETs*

In MANETs, network partitioning often occurs because mobile hosts act as a network router by themselves. Therefore, dynamic quorum systems have been considered as an approach suitable for consistency management in MANETs.

Several methods based on quorum systems have been proposed for MANET [10], [13], [14], [17]. In [14], the authors assumed an environment where a limited number of mobile hosts are chosen as replica holders and proposed a few methods in which the consistency is maintained by employing a dynamic quorum system. In [17], the authors extended the methods proposed in [14] by applying a *probabilistic quorum system* [18]. However, the methods proposed in [14] and [17] aim to roughly maintain the consistency in the entire network, but cannot completely maintain it.

In [10], we proposed the LC and GC protocols in which consistency among data operations is maintained based on a dynamic quorum system in a specific region where the entire area is divided into multiple regions. In the methods proposed in [14], [17] and the LC protocol, a flat (non-hierarchical) dynamic quorum system based on the condition, $|QR|+|QW|>P$, is used. In the GC protocol, a hierarchical (two levels) dynamic quorum system is used. In [13], we extended the GC protocol by using a grid quorum system at the higher level among proxies of regions to further reduce message traffic for protocol execution.

All the conventional flat quorum systems and the lowest level in all the hierarchical quorum systems adopt dynamic quorums based on the condition, $|QR|+|QW|>P$. Here, the results of simulation experiments presented in [10] showed that the number of replica holders much affects the performance of quorum systems. However, there have been no studies that fully investigate the impact of the number of replica holders and which nodes should replicate each data item.

III. 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 a flat area and not divided into sub-regions. We can extend to an environment where the area is divided into multiple sub-regions and the consistency is managed hierarchically by selecting a particular mobile host as the proxy of each region and construct quorums consisting of the proxies like GC presented in [10]. Details of the system model are as follows:

- We assign a unique *host identifier* to each mobile host in the system. The set of all mobile hosts in the system is denoted by $\boldsymbol{M} = \{M_1, M_2, \cdots, M_m\}$, where m is the total number of mobile hosts and M_i $(1 \leq i \leq m)$ is a host identifier. Each mobile host moves freely.
- Mobile hosts communicate with others using wireless communication. Messages and data items are exchanged between mobile hosts using an underlying routing protocol. We do not restrict the routing protocol and any existing protocols can be applied to our assumed system model.
- 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 $\boldsymbol{D} = \{D_1, D_2, \cdots, D_n\}$, where n is the total number of data items and D_j $(1 \leq j \leq n)$ is a data identifier.
  We assume that the size of a data item is much larger than that of a control message such as a request for quorum construction and the acknowledgment. Therefore, we should consider not only the reduction of message traffic for consistency management but also the reduction of data traffic for data operations.
  For the purpose of simplicity, we assume that all data items are of the same size, $|D|$.
- For each data item D_j, a fixed number (k_j) of mobile hosts $(M_{r_j1}, \cdots, M_{r_jk_j}$, which we call *replica owners*) are statically chosen as nodes having the right to replicate D_j, i.e., candidates involved in quorums for D_j.
  A replica owner of D_j does not have to always have a replica of D_j. However, it has to maintain the time stamp (version) of D_j independently of whether or not it actually has a replica of D'_j.
- Each mobile host 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 also assume that the frequencies of read and write operations, R_{ij} and W_{ij}, to each data item D_j from each mobile host M_i are known, and do not change. In a real environment, these access frequencies can usually be known by recording the logs of operation requests at each host.
- 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 the entire network. This assumption is based on the fact that many MANET applications require only simple transactions.
- We assume memory available at mobile hosts is large enough for creating replicas of all data items in the MANET.
- Each mobile host knows all mobile hosts having the right to replicate each data item in the entire network.
  This assumption can be achieved in real situations by informing all mobile hosts of this information at the configuration phase of the MANET. Even when a new host joins the MANET, it can get the information from another host that already participates in the MANET.

IV. Data Processing and Dynamic Quorum Construction

In this section, we present data processing and quorum construction methods assumed in this paper. We first describe the basic behavior of mobile hosts. Then, we present details of the data processing and quorum construction methods.

A. Basic Behavior

As mentioned in section III, for each data item D_j, a fixed number (k_j) of mobile hosts $(M_{r_j1}, \cdots, M_{r_jk_j})$, i.e., replica owners, are statically chosen as nodes having the right to replicate D_j. The value of k_j and how to choose k_j mobile hosts affect the system performance such as transaction success ratio (data availability) and traffic.

We adopt a dynamic flat quorum system, in which the quorum size for a read operation on data item D_j, $|QR_j|$, and that for a write operation, $|QW_j|$, are determined where the condition $|QR_j| + |QW_j| > k_j$ is satisfied.

In our previous work [10], it is shown that a write operation produces a large traffic for performing the operation on all the replicas held by mobile hosts in the write quorum, i.e., it requires transmissions of a data item of the latest version to all the nodes in the write quorum. However, in a quorum system, a write operation is not necessarily performed on all the replicas held by mobile hosts in the write quorum, but the consistency can be maintained by only performing the operation on some mobile hosts in the write quorum and informing the other hosts of the information on the time stamp (version) of the latest write operation and the replica holders of the latest version. Thus, we assume that a write operation is performed only on replicas held by h_j mobile hosts $(h_j \leq |QW_j|)$ in the write quorum and the others store the information on the time stamp and the h_j nodes. This might be effective for reducing traffic for write operations.

In the following, we describe the basic behavior of mobile hosts.

1) Phase (i): Lock Request: When a mobile host issues a request for a read (write) operation on data item D_j, it tries to set *read (write) locks* to arbitrary $|QR_j|(|QW_j|)$ replicas held by replica owners of data item D_j. If $|QR_j|(|QW_j|)$ is 1 and the request-issued node holds a replica of D_j, it sets the lock to its holding replica. Otherwise, it multicasts a lock request to all the replica owners. Each replica owner that received the request sets the lock to its holding replica of D_j and sends back a reply to the request-issued node. Here, note that it can happen that a replica owner does not have a replica of D_j

because a write operation is not necessarily performed to all replicas held by the replica owners in the write quorum. In such a case, the lock is virtually set at the replica owner.

As for a read operation, the reply contains the information on the version (time stamp) of the replica in either case the replica owner has a replica or not. It also contains the flag to represent whether the node has a replica (1) or not (0). If the replica owner does not has a replica, the reply contains the list of replica owners having the latest version.

If the request-issued node succeeds to set equal to or more than $|QR_j|(|QW_j|)$ locks, i.e., it received replies from equal to or more than $|QR_j|(|QW_j|)$ replica owners, the procedure goes to the next phase. Otherwise, the request fails.

2) Phase (ii): Data Operation: In this phase, the request-issued node performs the data operation on replicas to which the locks are set in the previous phase.

Read operation:
As for a read operation, if the request-issued node holds a replica of the latest version (it can be known from the information on the version attached in the query reply), it performs the read operation on its own replica, and then, the procedure goes to the next phase. Otherwise, if some nodes hold a replica of the latest version, the request-issued node unicasts a data transmission request to the closest node (with the shortest hopcount) having the latest version. Otherwise, i.e., no one holds the latest version but some nodes know replica owners having the latest version, the request-issued node randomly chooses one of the replica owners and unicasts a data transmission request to the node. This repeats until the request-issued node finds an accessible replica owner.

The replica owner that received the data transmission request transfers its holding replica. Then, the request-issued node sends back the acknowledgment to the sender node and performs the read operation. The procedure goes to phase (iii).

Write operation:
Let us denote QW'_j and k'_j as the set of replica owners that have sent a reply message in phase (i) and the number of such replica owners, respectively. The request-issued node transfers the replica of the new version to $\min(h_j, k'_j)$ nodes (suppose QW''_j) among nodes in QW'_j. Here, $\min(h_j, k'_j)$ is a function that returns the smaller value between h_j and k'_j. (h_j is the maximum number of nodes to which a write operation on D_j is actually performed.) Each node that received the replica of the new version replaces its holding replica with the received one, and sends back the acknowledgment to the request-issued node. If the node does not have a replica of D_j, it allocates the received replica on its free memory space. If the request-issued node receives the acknowledgment from all the $\min(h_j, k'_j)$ nodes, the procedure goes to phase (iii).

3) Phase (iii): Commit: In this phase, according to the result of phase (ii), the operation performed on the data item either commits or aborts.

Read operation:

The request-issued node that received a latest replica sends a commit message to the sender of the latest replica and other replica owners that sent a reply message in phase (i). Then, all the locks set to replicas for this operation are released and the procedure finishes. Even if the commit release message cannot reach all these nodes, it does not affect the database state, thus, the request-issued node can complete the operation.

Write operation:

In phase (iii) of a write operation, the request-issued node sends a commit message to all the nodes having replicas to which the write operation were performed in phase (ii) and other nodes that sent a reply message in phase (i). This message contains the list of replica owners having replicas to which the write operation was performed and the information on the time stamp of this write operation, i.e., the version of the latest replica. Then, each node that received the message sends back the acknowledgment to the request-issued node and releases the lock set to its holding replica. If the write operation was not performed on the replica held by the node, the node records the information on the list of the replica owners having the latest version and its time stamp.

If the request-issued node receives the acknowledgment from all the nodes having replicas to which the write operation was performed in phase (ii) and some replica owners that sent a reply message in phase (i) so that the total number of the nodes sending the acknowledgment is equal to or more than $|QW_j|$, the procedure finishes.

B. How to select k_j replica owners

In this clause, we describe how to select k_j replica owners for data item D_j. Actually, there are a large number of possible choices to achieve this. In this paper, we adopt a simple and intuitive manner where for each data item D_j, k_j mobile hosts with the highest access frequencies to D_j are chosen as the replica owners. Specifically, the following criterion G_{ij} is calculated for all mobile hosts.

$$G_{ij} = R_{ij} + W_{ij}. \tag{1}$$

Equation (1) represents how many data operations can be performed locally. Therefore, k_j mobile hosts with the highest G_{ij} are chosen as the replica owners for D_j.

Please note that this selection process is generally conducted once at the configuration phase of the MANET or fairly infrequently only when the access characteristics of nodes drastically change. This is because we assume that all nodes in the MANET know the replica owners for all data items, thus, changing replica owners requires the notification of the change to all the nodes and its overhead is very high.

C. How to select h_j nodes to which a write operation is performed

In this clause, we describe how to select h_j mobile hosts to which a write operation is performed. Unlike replica owners, these h_j nodes can be chosen more flexibly and dynamically when a write operation is performed. This is because request-issued nodes do not need to know in advance which mobile hosts have been performed the latest write operation but it knows these nodes after phase (i) of the write operation. However, the strategy to determine these h_j mobile hosts much affects the performance, e.g., data availability and traffic. Thus, we should carefully choose the strategy. In this paper, we adopt two different strategies, *AF (Access Frequency)* and *DIST (Distance)*.

1) AF: AF aims to reduce traffic for data transmissions for future data operations. Specifically, for each candidate node, the request-issued host calculates the gain of data traffic reduction by equation (1), i.e., G_{ij}. AF chooses h_j (more precisely, $\min(h_j, k'_j)$) nodes with the highest G_{ij} among all nodes in QW'_j. It should be noted that to achieve this method, the information on the access frequencies, W_{ij} and R_{ij}, should be added in the query reply.

2) DIST: DIST aims to reduce traffic for data transmissions for the current data operation by choosing closer mobile hosts for performing the write operation. Specifically, DIST checks the hopcounts between the request-issued host and all nodes in QW'_j, which can be obtained from the reply messages received in phase (i). Then, it chooses h_j mobile hosts with the smallest hopcounts from it, to which the write operation is performed.

V. Simulation

In this section, we show simulation results to investigate the impact of quorum construction on the performance in terms of data availability and traffic.

A. 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 equipped with mobile terminals with a wireless communication facility such as Bluetooth, wireless LAN, and Zigbee. In the experiments, we assume that a unit of simulation time corresponds to 5 [s] in a real environment. In the following, we present the details of the simulation model.

Mobile hosts exist in an area of $X \times Y$ [m^2]. Here, ratio $X : Y$ is kept to $3 : 4$. In our experiments, X is changed as a variable parameter in the range from 100 [m] to 500 [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 100. ($M = M_1, \cdots, M_{100}$). The number of data items in the entire network is 500 ($D = D_1, \cdots, D_{500}$). Each peer moves according to the random waypoint model [3], where each host selects a random destination in the whole area. The pause time and the maximum movement speed are set as 0 and 2 [m/s], respectively, assuming that peers move in a walking speed.

The communication range of each mobile host is a circle with radius 50 [m]. To remove the impact of underlying network protocols, we simply assume that every message and data transmission is routed via the shortest path from the source to the destination. Moreover, even in a multicast transmission, e.g., lock request, a message or data item is separately transmitted to each of the multicast members via the shortest path, i.e., a multicast transmission consists of separate unicast transmissions. While the protocol for processing data operations presented in subsection IV-A adopt 2PC (two-phase commitment) in phase (iii), the possible approaches for this aim depend on the implementation, thus, we just count the hopcount for one round of message transmissions.

Read and write frequencies of each peer to data items are 0.02 [1/s] and 0.002 [1/s], respectively. The read/write probability, q_{ij}, from mobile host M_i to data item D_j follows the Zipf distribution [20] and is expressed by the following equation:

$$q_{ij} = \frac{j'^{-0.6}}{\sum_{m=1}^{500} m^{-0.6}}, \quad \text{where} \quad j' = \begin{cases} j-i+1 & (j \geq i) \\ 501+j-i & (j < i) \end{cases} \tag{2}$$

Equation (2) shows that all mobile hosts have different access characteristics where M_i access D_i most frequently among all data items.

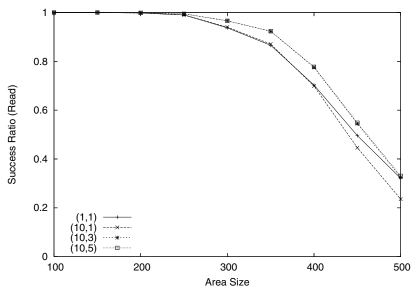
k_j (the number of replica owners) and h_j (the number of mobile hosts to which a write operation is actually performed) are respectively set to the same values for all data items, while we examine various cases for their values.

In the simulations, we measured 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 [s]). 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 don't 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.

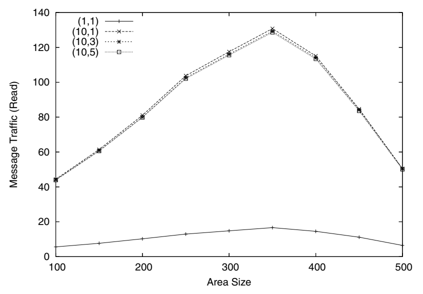
We assume that the protocol execution finishes within a unit of simulation time, i.e., 5 [sec]. This is usually true in a real environment, because one round of message exchanges in the entire network takes less than one or two seconds, and thus, the entire procedure that requires three rounds of messages 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.

B. Results

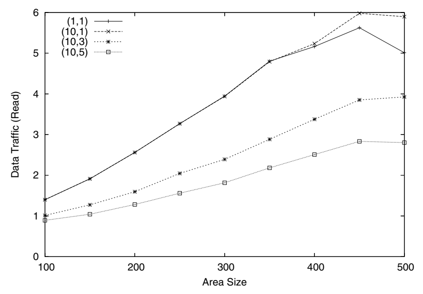
Figures 1 and 2 show results of the simulations where AF and DIST are respectively adopted for selecting h_j hosts. In both figures, graphs (a) to (f) and graphs (g) to (l) respectively



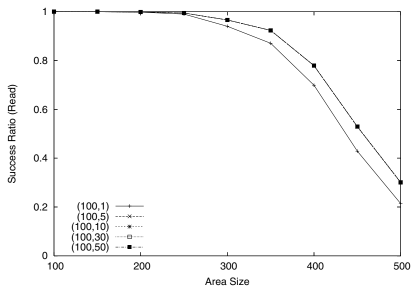
(a) success ratio (read, $k_j = 1, 10$)



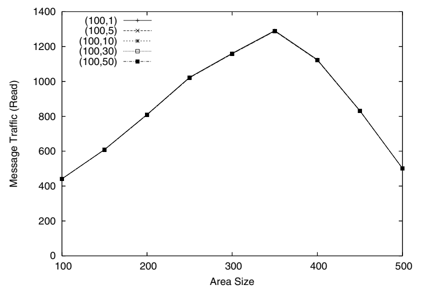
(b) message traffic (read, $k_j = 1, 10$)



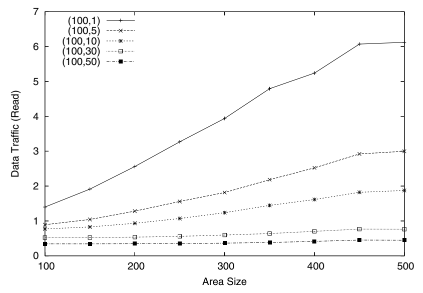
(c) data traffic (read, $k_j = 1, 10$)



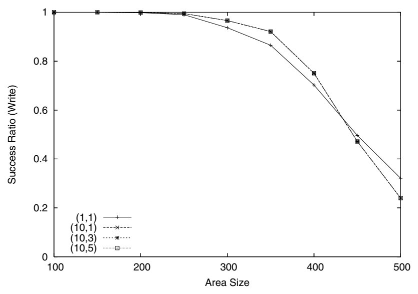
(d) success ratio (read, $k_j = 100$)



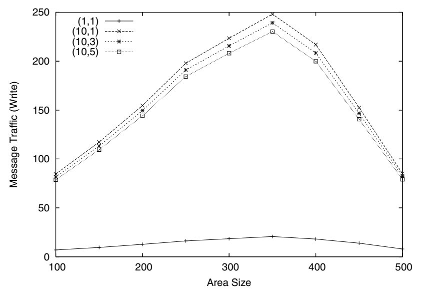
(e) message traffic (read, $k_j = 100$)



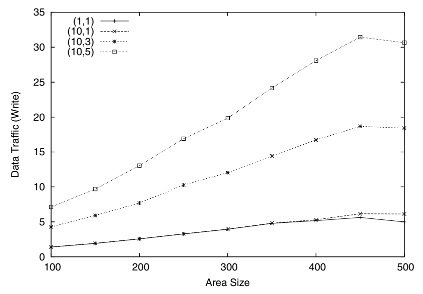
(f) data traffic (read, $k_j = 100$)



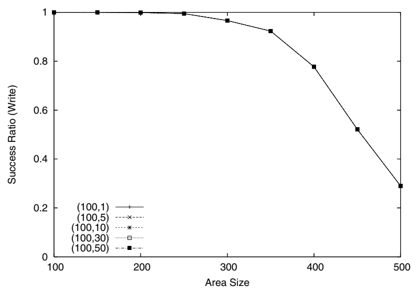
(g) success ratio (write, $k_j = 1, 10$)



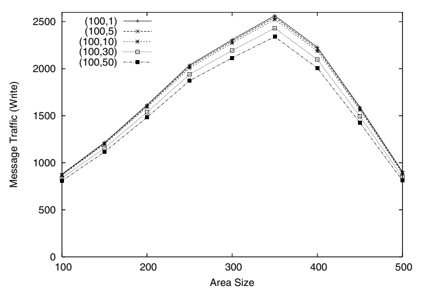
(h) message traffic (write, $k_j = 1, 10$)



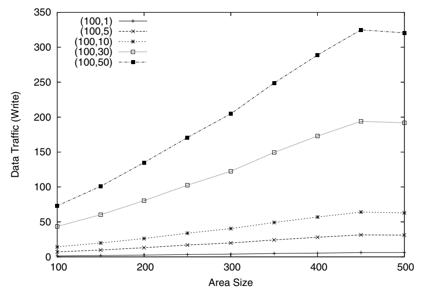
(i) data traffic (write, $k_j = 1, 10$)



(j) success ratio (write, $k_j = 100$)



(k) message traffic (write, $k_j = 100$)



(l) data traffic (write, $k_j = 100$)

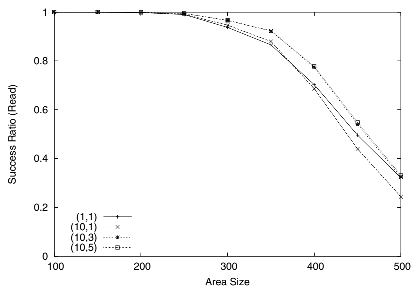
Fig. 1. Effects of the area size (AF)

correspond to the results for read and write operations. Graphs (a) to (d) and (g) to (i) show the cases where $k_j = 1$ and $k_j = 10$ while the rests show the cases where $k_j = 100$. In all graphs, each line is labeled as (k_j, h_j).

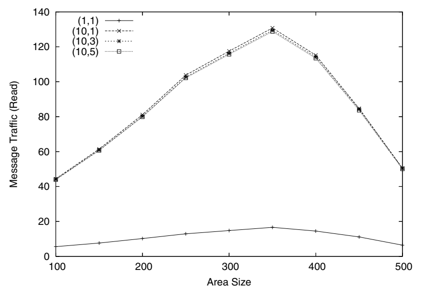
1) Overview of the results: In all cases, as the area size gets larger, the success ratios for both read and write operations get lower because the connectivity between mobile hosts gets lower.

The message traffic first gets higher for both read and write operations and then gets lower from a certain point ($X = 350$) as the area size gets larger. The message traffic is low when the area size is small because the network is dense and hopcounts between mobile hosts are small. It is low when the area size is very large because the network is sparse and mobile hosts connect to only a small number of other nodes.

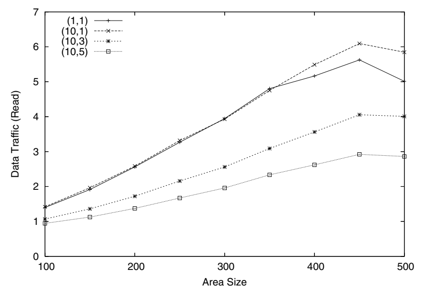
As the area size gets larger, the data traffic basically gets higher because of the increase of hopcounts between mobile hosts. It becomes almost constant or gets slightly lower when the area size is very large. This is because the network is partitioned into several small partitions and operation requests



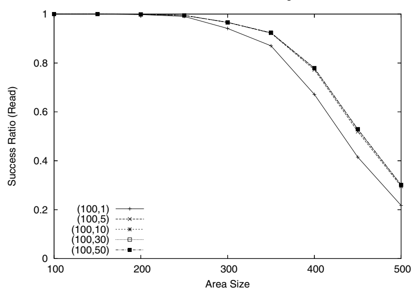
(a) success ratio (read, $k_j = 1, 10$)



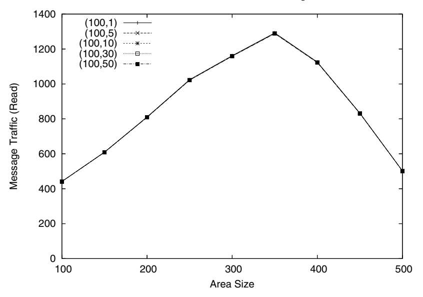
(b) message traffic (read, $k_j = 1, 10$)



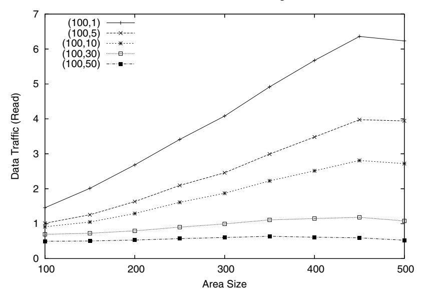
(c) data traffic (read, $k_j = 1, 10$)



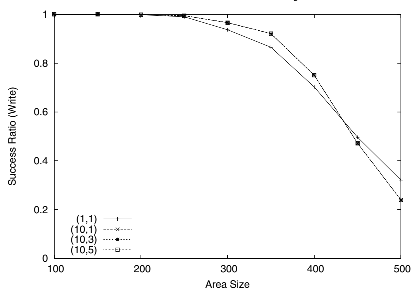
(d) success ratio (read, $k_j = 100$)



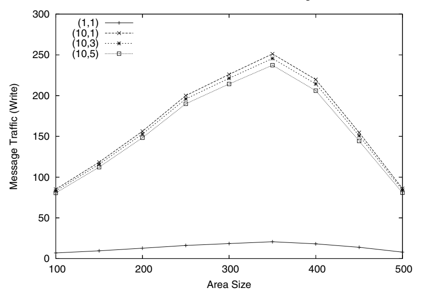
(e) message traffic (read, $k_j = 100$)



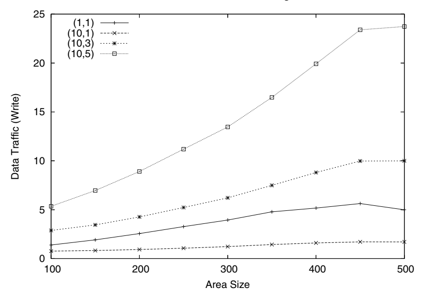
(f) data traffic (read, $k_j = 100$)



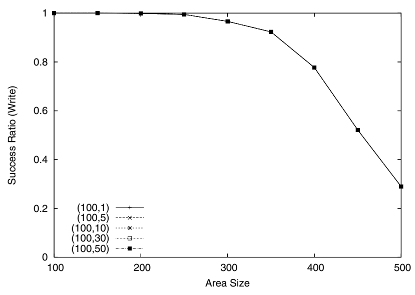
(g) success ratio (write, $k_j = 1, 10$)



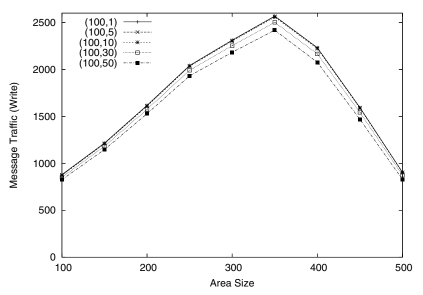
(h) message traffic (write, $k_j = 1, 10$)



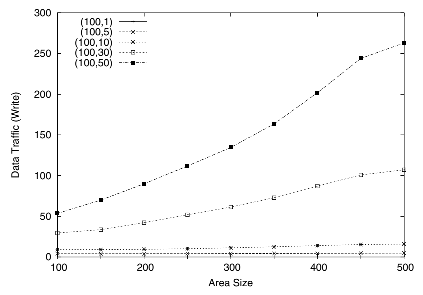
(i) data traffic (write, $k_j = 1, 10$)



(j) success ratio (write, $k_j = 100$)



(k) message traffic (write, $k_j = 100$)



(l) data traffic (write, $k_j = 100$)

Fig. 2. Effects of the area size (DIST)

succeed only when the request-issued hosts luckily connect to an enough number of replica owners in the partition to which they belong.

2) Comparison between AF and DIST: Comparing the results for AF and DIST in Figures 1 and 2, there are almost no differences in the success ratio and the message traffic. As for the data traffic, AF produces smaller data traffic for read operations because request-issued hosts hold a latest replica with a higher probability. On the other hand, DIST produces smaller data traffic because it performs a write operation to h_j closest hosts. This result shows that the strategy for selecting h_j hosts to which a write operation is performed affects the performance in terms of traffic for data operations.

3) Impact of k_j: From the results in Figures 1 and 2, we can see that k_j directly affects the message traffic, i.e., the larger k_j is, the larger the message traffic is.

In terms of success ratio, when the area size is very large, i.e., the network is very sparse, the case where k_j (and also h_j) is set as 1 shows higher success ratio than the cases where k_j is set as 10 and 100 and h_j is set as 1. This is because mobile hosts in the partition to which the replica owner belong can perform data operations where $k_j = 1$, even if the partition size is very small. On the contrary, the case where $k_j = 1$ shows lower success ratio when the network is dense. This is because mobile hosts cannot perform any data operations even when only the replica owner is not accessible, i.e., the flexibility and the robustness to the network topology change are low.

In terms of data traffic, larger k_j shows lower data traffic for write operations because there are more candidates for choosing the closest hosts to which a write operation is actually performed.

From these results, we can confirm that k_j affects the performance especially in terms of message traffic.

4) Impact of h_j: From the results in Figures 1 and 2, we can see that h_j directly affects the data traffic, i.e., the larger h_j is, the larger the data traffic for write operations is and the smaller that for read operations is.

The case where h_j is set as 1 and k_j is set as 10 or 100 shows lower success ratio than the cases where h_j is larger than 1. This is due to the same reason as the case where $k_j = 1$ in the previous clause. On the other hand, h_j does not much affect the message traffic.

VI. Conclusions

In this paper, we have investigated the impact of quorum construction on the system performance through simulation studies. In the simulations, we changed the number of mobile hosts that replicate data items and which hosts replicate each data item and examined the impact on the system performance in terms of data availability (success ratio) and traffic (message and data traffic).

From the simulation results, we have confirmed that the sizes of quorums, the number of replica holders, and the strategy for determining replica holders affect the performance. The knowledge obtained from our simulation results will be a good guideline for constructing efficient quorums in MANETs.

Acknowledgments

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

References

[1] D.J. Baker, J. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing, mobile, radio network," *Proc. IEEE ICC'82*, pp.2F6.1-2F6.5, 1982.

[2] D. Barbara and H. Garcia-Molina, "The reliability of vote mechanisms," *IEEE Trans. on Computers*, Vol.36, No.10, pp.1197-1208, 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. Mobicom'98*, pp.159-164, 1992.

[4] G. Cao, L. Yin, and C.R. Das, "Cooperative cache-based data access in ad hoc networks," *IEEE Computer*, Vol.37, No.2, pp.32-39, 2004.

[5] S. Cheung, M.H. Ammar, and M. Ahamad, "The grid protocol: a high performance scheme for maintaining replicated data," *IEEE Trans. on Knowledge and Data Engineering*, Vol.4, No.6, pp.582-592, 1992.

[6] M.K. Denko, E. Shakshuki, H. Malik: "A mobility-aware and cross-layer based middleware for mobile ad hoc networks, *Proc. AINA 2007*, pp.474-481, 2007.

[7] 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.

[8] T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," *Proc. IEEE Infocom 2001*, pp.1568-1576, 2001.

[9] T. Hara, and S.K Madria: "Data replication for improving data accessibility in ad hoc networks," *IEEE Trans. on Mobile Computing*, Vol.5, No.11, pp.1515-1532, 2006.

[10] T. Hara, and S.K Madria: "Consistency management strategies for data replication in mobile ad hoc networks," *IEEE Trans. on Mobile Computing*, Vol.8, No.7, pp.950-967, 2009.

[11] T. Hara: "Quantifying impact of mobility on data availability in mobile ad hoc networks," *IEEE Trans. on Mobile Computing*, Vol.9, No.2, pp.241-258, 2010.

[12] M. Ikeda, L. Barolli, G. De Marco, T. Yang, A. Durresi, F. Xhafa: "Tools for performance assessment of OLSR protocol," *Mobile Information Systems*, Vol.5, No.2, pp.165-176, 2009.

[13] A. Kanzaki, S. Sawai, M. Shinohara, T. Hara, and S. Nishio: "Quorum-based consistency management for data replication in mobile ad hoc networks,", *Proc. Int'l Workshop for Ubiquitous Networking and Enablers to Context-Aware Services*, pp.357-360, 2008.

[14] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information dissemination in partitionable mobile ad hoc networks," *Proc. SRDS'99*, pp.4-13, 1999.

[15] A. Kumar, "Hierarchical quorum consensus: A new algorithm for managing replicated data," *IEEE Trans. on Computers*, Vol.40, No.9, 1991.

[16] W. Luk, and T. Wong, "Two new quorum based algorithms for distributed mutual exclusion," *Proc. IEEE ICDCS'97*, pp.100-106, 1997.

[17] J. Luo, J.P. Hubaux, and P. Eugster, "PAN: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems," *Proc. ACM MobiHoc 2003*, pp.1-12, 2003.

[18] D. Malkhi, M.K. Reiter, and A. Wool, "Probabilistic quorum systems," *Information and Computation*, Vol.170, No.2, pp.184-206, 2001.

[19] P. Padmanabhan, L. Gruenwald, A. Vallur, M. Atiquzzaman, "A survey of data replication techniques for mobile ad hoc network databases," *VLDB Journal*, Vol.17, No.5, pp.1143-1164, 2008.

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