Mobile Information Systems 2 (2006) 193–209
IOS Press

Replica dissemination and update strategies in cluster-based mobile ad hoc networks

Mieso K. Denko* and Hua Lu
Department of Computing and Information Science, University of Guelph, Guelph, Ontario, Canada, N1G 2W1

Abstract. A mobile ad hoc network (MANET) is a collection of wireless mobile nodes that forms a temporary network without the aid of a fixed communication infrastructure. Since every node can be mobile and network topology changes can occur frequently, node disconnection is a common mode of operation in MANETs. Providing reliable data access and message delivery is a challenge in this dynamic network environment. Caching and replica allocation within the network can improve data accessibility by storing the data and accessing them locally. However, maintaining data consistency among replicas becomes a challenging problem. Hence, balancing data accessibility and consistency is an important step toward data management in MANETs. In this paper, we propose a replica-based data-storage mechanism and undelivered-message queue schemes to provide reliable data storage and dissemination. We also propose replica update strategies to maintain data consistency while improving data accessibility. These solutions are based on a clustered MANET where nodes in the network are divided into small groups that are suitable for localized data management. The goal is to reduce communication overhead, support localized computation, and enhance scalability. A simulation environment was built using an NS-2 network simulator to evaluate the performance of the proposed schemes. The results show that our schemes distribute replicas effectively, provide high data accessibility rates and maintain consistency.

Keywords: Mobile ad-hoc networks, data management, mobile computing, wireless networks, replication, caching

1. Introduction

A MANET is characterized by highly dynamic, fully self-organized, and unpredictable topological changes. When routes do not exist, packet transmission is deferred until a fresh path is found through a route discovery mechanism in on-demand routing or is recomputed using proactive routing protocols. Although a number of network layer protocols have been designed for MANETs, little work appears in the literature in regard to the effect of this dynamic network topology on higher-level services such as data storage and access.

In recent years, a number of data-storage and message-delivery mechanisms have been proposed for infrastructure-based wireless networks. In [23], an active message-based delivery system was proposed for data transmission in wireless networks that experience frequent link failures. This active message transmission mechanism is based on mobile agents and allows message transmission in networks that suffer from frequent link failures. In [19], a similar message transmission mechanism for disconnected ad-hoc networks was proposed. This mechanism, which is based on the concepts of mobile agents and active messages, considers two scenarios, namely: (a) communication with full knowledge of the host

*Corresponding author. Tel.: +1 416 798 1331 ext. 6088; E-mail: denko@cis.uoguelph.ca.

movement and (b) communication without the full knowledge of the host movement. The scenario regarding the full knowledge of host movement is valid only when the node trajectory is known. In the scenario without the full knowledge of host movement, a location-update mechanism is used for maintaining location information.

Caching and replication techniques were proposed in [3,7,12,21,27] to improve data accessibility. These caching schemes can be classified as a reactive caching scheme, pro-active caching scheme or hybrid-caching scheme. For example, in [7] a reactive mechanism that locates cache proxies in MANETs was proposed. It introduces search dissemination techniques, including pure flooding and several variations of a flooding algorithm for proxy lookup. In a proactive caching scheme, copies of the data item are replicated on those nodes to which requests are likely to be issued even before the requests are routed through those nodes. Several replica allocation mechanisms were also developed for MANETs. For example, three replica allocation methods that consider static access frequency, dynamic access frequency, and neighborhood and dynamic connectivity-based grouping [13] were proposed. This was used to improve data accessibility by replicating the data items shared by mobile hosts. The data item that will be replicated is chosen based on the access frequency from the mobile hosts, the status of the network connection, and the time remaining until each item is next updated. The work was extended in [14,15] by taking the data update into account. In [27], a protocol was proposed that uses nodes that move between the neighborhoods of the source and destination nodes to act as carriers of the message. The performance of this mechanism depends on the selection of carrier nodes. Also, the protocol is of little use when there are no carrier nodes available between the source and the destination. In [10], a peer-to-peer data dissemination mechanism that uses the Tornado coding technique [2] was proposed. The basic idea behind Tornado coding is that the data provider encodes the original file in small pieces and the data receiver re-assembles the original file when the receiver receives sufficient data segments from the data provider.

In [22], a probabilistic quorum system for Ad-hoc Networks was proposed for providing reliable data storage with concurrent read/write access in MANETs. This gossip-based protocol for quorum access assumes purely random mobility patterns and dense networks. It was designed to provide reliable data storage for small data objects. Although the reliability and robustness of the mechanism was demonstrated through simulation, the scalability of the protocol was not considered. In [26], a skip-copy method was proposed to improve accessibility to data that are used frequently by nodes close to the location. This method is based on replicating data objects, and the replicas are sparsely distributed around the location where they are associated. Replicas can be rearranged or re-disseminated when they are forwarded to the requesting node. However, this scheme does not take data update into account. In [10], a peer-to-peer data dissemination mechanism that uses the Tornado coding technique was proposed. The application of Tornado coding for data dissemination in MANETs was investigated. In [11], the problem of replica synchronization in autonomous grid systems was discussed. A novel replica synchronization protocol known as Grid Replica Access Protocol (GRAP), which operates in heterogeneous grid architecture, was proposed. GRAP uses the grid middleware's metadata service and timestamps to synchronize the replicas among distributed sites. An extended version of GRAP, Contingency-GRAP (C-GRAP), which deals with multiple networks partitioning, was also presented. The C-GRAP protocol allows transactions to proceed even if the quorum cannot be obtained.

We argue that most existing data storage and dissemination schemes have limitations, particularly in terms of reliability, processing overhead, communication overhead, and scalability. For example, running mobile agents for the mechanisms proposed in [19,25] can result in high computation and communication overhead. Also, security is another potential problem due to the inherent security challenges in mobile

agents as well as MANETs. We propose to tackle the data management problem by using an undelivered message queue mechanism, based on our earlier work [20], to ensure reliable replica, distribution, and consistent updates even when the nodes are temporarily disconnected from the network. The replica update strategies proposed in [21] are integrated with the two replica dissemination algorithms to achieve consistency in each cluster.

In this paper, we propose a novel data-management scheme for replica storage, dissemination, and updates. The replica dissemination mechanism maintains an undelivered message queue to provide reliable data storage and delivery in clustered MANETs. In our schemes, a data item sent from a data sender to a data receiver will not be discarded when the receiver is temporarily away. Instead, the data item will be stored in an undelivered-message queue (UMQ) in a cluster head (CH) and delivered as soon as the receiver rejoins the cluster or the network. The main contributions of this paper are the following: (a) We have proposed two novel algorithms for replica dissemination. (b) We have integrated two replica update algorithms with the replica dissemination algorithms to provide an integrated data management scheme for MANETs. The data management scheme was evaluated in a simulation environment under various parameter settings.

The rest of this paper is organized as follows. Section 2 presents the architecture of the proposed data management scheme. Section 3 presents the proposed replica storage and dissemination mechanisms. Section 4 presents the replica update mechanisms. Section 5 presents results of the performance evaluation, and finally, Section 6 concludes the paper by presenting conclusions and suggestions for future research work.

2. The architecture of the proposed scheme

2.1. Problem specification

MANETs are expected to play dominant role in future civilian and military applications by facilitating information access where no wired infrastructure exists to support communication. These application scenarios can range from low mobility to vehicular-speed applications. They can also be used where the creation of wired or wireless infrastructure is not feasible for historical or financial reasons. Two simple sample scenarios are described below.

Scenario 1: Consider a historical multi-story building where no wired infrastructure exists for communication and where mobile nodes form an ad-hoc network. Assume that there is a cluster of nodes on every story of the building. There can be several clusters in the entire building. Nodes move freely within a cluster in any direction. However, in this scenario, if a node leaves one cluster and joins another cluster, the node can only move upwards to a higher story or downwards to a lower story of the building through elevators or stairwells. When a mobile node moves inside a building, the node can be disconnected from the cluster or the network.

Scenario 2: Consider firefighters carrying mobile devices while performing a rescue task in a building complex. Some try to rescue residents trapped by fire while others fight the fire. Firefighters periodically send messages to update their location or operational status, or to provide information or shared data regarding subjects such as the number of people who are trapped, exploding hazard sources, dangerous chemical substances, and other information of use to the officers. Moreover, information such as data about people and property in the area may be recorded and shared among teams of medics, reporters, and individual firefighters. Officers need to collect and disseminate data with firefighters and then coordinate the whole task. Both officers and other team members can lose network connections temporarily (due to mobility or link failures) without leaving the network completely.

2.2. Clustering architecture

The main purpose of clustering is to provide a more efficient utilization of the network resources through spatial reuse to enhance availability, reduce overhead by providing services locally, and provide scalable architecture [5,9]. The choice of a clustering algorithm affects the stability of clusters. Several algorithms have been proposed for clustering nodes in wireless networks. Since a cluster head (CH) is expected to function for longer durations once elected, we use a mobility-based clustering algorithm [5] for cluster formation. In this algorithm, a node is elected as a CH only if the node's mobility rate is below a certain threshold. In the case of a tie, the node with the lowest ID is chosen.

Every CH maintains a table that contains the IDs of all nodes in its cluster, IDs of CHs in other clusters, border nodes, link status, and mobility information. The CHs are coordinators in each cluster and store shared information. A border node is a node that can communicate with neighboring clusters. Each CH knows how many CHs are available in the network. Every node periodically broadcasts a "hello" message to maintain information about the CH and its neighbors, link status, and mobility rate. The mobility rate measures the relative mobility of a node with respect to its neighbors. We consider a relatively dense network in our study. Clusters can merge or split based on a predefined merge and split threshold. When a new CH is elected, it sends a message with its own ID informing other CHs that a new cluster head has been elected and the original CH is replaced. After the other CHs receive this message, they record the new CH in corresponding tables, delete the original CH, and then send a reply message with their own IDs.

2.3. Description of proposed scheme

Several correctness and consistency control mechanisms exist in literature. The token-based mechanism is not appropriate for ad hoc networks since the token may be lost due to the dynamic nature of the network. The quorum-based mechanisms incur high communication overhead as the algorithm must be executed too often to run them. Another approach uses the read one/write all algorithm. However, this may result in an incorrect execution when two updates on the same data item are invoked through two different nodes simultaneously. Therefore, in our approach we use the update lock mechanism to handle data consistency.

We studied replica caching and data update strategies in a clustered, wireless ad hoc network environment. In our proposal, cluster heads are elected among relatively more stable nodes in an ad hoc network using the cluster head election algorithm proposed in [5]. However, any similar algorithms such as [8] or [24] can be used to elect the cluster head. We make the following assumptions concerning the system environment.

The network is divided into several one-hop clusters in which in each cluster a node could be present in one of three different statuses: cluster head, ordinary node, or gateway. A gateway is a node that can communicate with two clusters directly. Each node maintains and updates neighbor information by exchanging "hello" messages periodically.

We assign a unique data descriptor to each data item. Each node has its own data items and replicas (cached data item) from other nodes. The set of original data items owned by a node is denoted by $D = \{\{D_{11}, D_{12}, \ldots, D_{1n}\}, \{ D_{21}, D_{22}, \ldots, D_{2n} \}, \ldots, \{ D_{m1}, D_{m2}, \ldots, D_{mn} \}\}$, where m is the total number of mobile nodes in MANET, n is the total number of data items in a mobile node, and D_{ij} ($1 \leqslant$ i $\leqslant$ m, $1 \leqslant$ j $\leqslant n$) is a data descriptor. Similarly, the set of replicas is denoted by $R = \{\{ R_{11}, R_{12}, \ldots, R_{1r} \}, \{ R_{21}, R_{22}, \ldots, R_{2r} \}, \ldots, \{ R_{m1}, R_{12}, \ldots, R_{mr} \}\}$, where m is the total number of mobile nodes in MANET, and r is the total number of replicas cached in a mobile node.

Each data item is associated with an update lock, which is required by a node in order to update this data item. The set of update locks is denoted by $L = \{\{ L_{11}, L_{12}, \ldots, L_{1n} \}, \{ L_{21}, L_{22}, \ldots, L_{2n} \}, \ldots, \{ L_{m1}, L_{m2}, \ldots, L_{mn} \}\}$, where m is the total number of mobile nodes in MANET, n is the total number of node's own data items, and L_{ij} ($1 \leqslant i \leqslant m$, $1 \leqslant$ j $\leqslant n$) is an update lock.

Each replica item is associated with an expiry time and is set to the current time of the data owner plus the update period. Every node purges its local cache periodically to remove the expired replicas. The replica becomes invalid after its expiry time.

3. Replica storage and dissemination

We consider two types of data in our network. The first type of data is shared data, which is used by the mobile nodes. The size and contents of these data differ depending on the application environment. For instance, data may be a map used by a disaster recovery team, tourists, or military personnel. Data may include files shared among conference participants, for example. Full replicas of shared data are stored in all the CHs. The second type of data is the messages that are used by communicating nodes. These transmitted messages can be queries, replies, or updates to the shared data.

The key issues in data management are availability, reliability, and consistency. Availability requires that the data be accessible when needed with minimum disruption. Since MANETs experience frequent topological changes, data availability and the need to ensure reliable transmissions pose challenging problems. Moreover, users need to access data despite frequent disconnections with minimum query-retrieval operations and minimum energy and bandwidth requirements.

Availability can be achieved through data replication, but data inconsistencies may occur in networks that suffer from frequent network partitions. A related problem in MANETs is transmission reliability, which is impaired either by mobility or reduced signal strength. Queries and reply messages should be transmitted with minimum error rates or higher packet delivery ratios, in order to provide the required quality of service.

3.1. Replica dissemination mechanisms

The existing data-dissemination mechanisms for the Internet and infrastructure-based wireless networks are not suitable for MANETs. The main reason for this is that they consider neither the dynamic network topology nor the limited available network resources. To increase the reliability of data transmission, we replicated the undelivered messages among all or a subset of clusters. Messages do not require updates, but they do require acknowledgements (ACK) upon successful delivery. Data replication can be implemented as a multicast, while updates can be either multicast or broadcast. When network partitioning occurs, shared data can still be accessed within the network within in each cluster, and messages are stored at CHs.

An essential component of our data-dissemination mechanism is the undelivered-message queue. The basic idea is that if a cluster head receives a message marked as an undelivered message, the CH will store the message in the undelivered-message queue (UMQ). When the disconnected node joins the MANET again, the CH will deliver the messages in the UMQ. When the message in the UMQ cannot be delivered within the Time to Live (TTL) period, the CH will send a negative ACK to the message sender and the message will be deleted.

We propose two different algorithms based on two implementations of the UMQ for message transmission: the Prediction-based Message Queue (PMQ) algorithm and the Distributed Message Queue (DMQ) algorithm. These algorithms are described in the following sections.

3.2. The prediction-based message queue (PMQ) algorithm

In the PMQ algorithm, if a message could not be delivered due to the disconnection of the destination node from the cluster, the message will be sent to the CH for storage in the CH's UMQ. The CH will invoke a prediction algorithm in order to find the cluster into which the disconnected node might have moved. The CH will then send a copy of that message to the CHs of selected clusters. Thus, the first level of location prediction is cluster prediction, while the second level is locating the mobile node itself. We will consider three scenarios to describe how the PMQ algorithm operates.

Scenario 1: Consider sending a message to a node, which is in its home cluster. For example, node N_i in cluster C_m sends a message to node N_j, which is in cluster C_n. In this case there will be no problem for N_j to get this message since the node is located in cluster C_n. When N_j receives the message, the node sends a positive ACK back to the message sender, N_i.

Scenario 2: Consider sending a message to a node that has moved to another cluster and may not return. For example, node N_i in cluster C_m sends a message to node N_j, which was expected to be in cluster C_n but has moved to another cluster. Under our PMQ algorithm, this message will be forwarded to the CH in cluster C_n. The CH in C_n will perform the prediction algorithm and multicast this message to CHs in clusters $\in S_t$ where S_t denotes the cluster set that contains this undelivered message. N_j will get this undelivered message when N_j joins the cluster $C \in S_t$. If N_j joins a cluster $C \notin S_t$, N_j cannot receive this message because the CH in its current cluster does not have this message in the UMQ. If a message in its UMQ cannot be delivered within the TTL period, the CH will send a negative ACK to the message sender.

Scenario 3: Consider sending a message to a node that is not currently in its home cluster but will rejoin later. Suppose node A in cluster 1 is disconnected both from the cluster and the MANET and rejoined cluster 1 after a certain period of time. Suppose also node B in cluster 2 sends a message to node A assuming that it is still in cluster 1. However, node A is out of the transmission range of any node in cluster 1, which means that this message cannot be delivered to node A. Under the PMQ algorithm, this message will not be discarded but will be forwarded to the CH in cluster 1 and then forwarded to the predicted set of clusters. When node A rejoins the cluster, the CH will send the node a copy of the messages in the UMQ.

The main advantages of the PMQ algorithm are its simplicity and the low communication overhead the algorithm incurs. However, this algorithm cannot guarantee the delivery of a message to a node that has moved to another cluster. We propose another algorithm, distributed undelivered-message queue (DMQ), to overcome this problem.

3.3. The distributed message queue (DMQ) algorithm

The DMQ algorithm is different from the PMQ algorithm discussed above in that when a CH receives an undelivered message, the DMQ stores the message in its UMQ and also sends a copy of this message to the other CHs. The shared data are stored in the CH as in the PMQ algorithm. Hence, the operations of CHs in the DMQ are similar to those in the PMQ algorithm, but the location-prediction module is not used in DMQ. Thus, in DMQ we have a set of globally unique UMQs in the MANET. The UMQ in every CH maintains all the currently undelivered messages.

There are three main operations involving a CH under PMQ: (a) place the message in the UMQ, (b) deliver the message when the destination is available, and (c) delete the message in the UMQ when the message's TTL expires. Using the DMQ algorithm, a mobile node will receive messages successfully in the scenarios described in the preceding section for PMQ. When the CH is disconnected from the cluster, a new CH will be elected.

4. Replica update mechanisms

While data replication and caching strategies improve data availability and reliability in frequently partitioned networks, maintaining correctness and consistency poses a challenge [4] in distributed systems. Due to node mobility, limited bandwidth, and low battery power, ad hoc networks are characterized by disconnected operation and weak connectivity, hence encountering more challenges than traditional distributed databases. Providing replica update strategies and a mechanism for data delivery after node reconnection or network partition recovery is a step forward to tackle the problem. In what follows, we will discuss the local update algorithm, which implements the data update in a cluster, and the global update algorithm, which maintains data consistency in the entire network.

Researchers have made different assumptions regarding network and data-related operations while proposing mechanisms for data dissemination and replica allocation in mobile and wireless networks. In [25], a data-replica allocation mechanism, which assumes unlimited memory space, was proposed. In [13], limited memory was assumed, but the data were not updated, so consistency was not a problem. To increase the availability of data, we replicated the shared data in the network. To reduce the overhead, we used the clustering architecture; the replicas of shared data are stored only at the CHs of each cluster. In our scheme, clusters are formed to minimize CH re-election or network partitioning. To maintain consistency, the shared data are updated. Upon network partitioning, we employ constrained updates by carrying out updates only in one partition [4]. This is achieved by allowing updates only in the partition with the larger number of clusters.

Upon the merger of the partitions, the replicas are synchronized so that all the CHs have the same data. This operation may incur overhead on mobile nodes but since the update is only made at the CHs, the effect is minimal. We minimized the data update as much as possible; however, if updates are required, time-based and event-based approaches are used. When there has been no event since the last update, we use a time-based update; otherwise, we use an event-based update. The updates are mainly made at the CHs and can be broadcast to other nodes in the cluster or to other CHs. An event may be a location change, new information, or location-dependent information for use by a group.

4.1. Replica distribution, data request, and updates

In our scheme, when data dissemination failure occurs, which could be when a replica request or update message is detected, the message in transit will be forwarded to a cluster head for further processing, instead of being retransmitted or a dropping of the packet occurring. Mobile nodes, situated in the same cluster, update the data item. The original data owner and its neighbors who hold the replica can invoke an update. This is reasonable in many MANET applications. Considering the case of a rescue task being carried out in a fire field, a new data item, such as the one indicating that a person has been found besieged on a particular story, is created in one cluster and then added to the database. Later on, this data item may be updated when more people are found or some people have already been moved out to a safe place. Firefighters in other clusters can read the data item but are not allowed to update.

We use a hybrid data request method in our scheme. A node proactively sends out a replica using the replica distribution procedure, and sends out replicas to current cluster heads. On the other hand, when the replica becomes invalid or no replica is available, a node can request a data item in a reactive way. Before a node issues a data item request, the node will first search its own cache to check whether the needed data item is there. If the cache does not have the data, the node will broadcast a data request message within the cluster. If the node does not receive a reply from its neighbors, the node then sends out a data request to the whole network. Any node that receives such a data request will then check

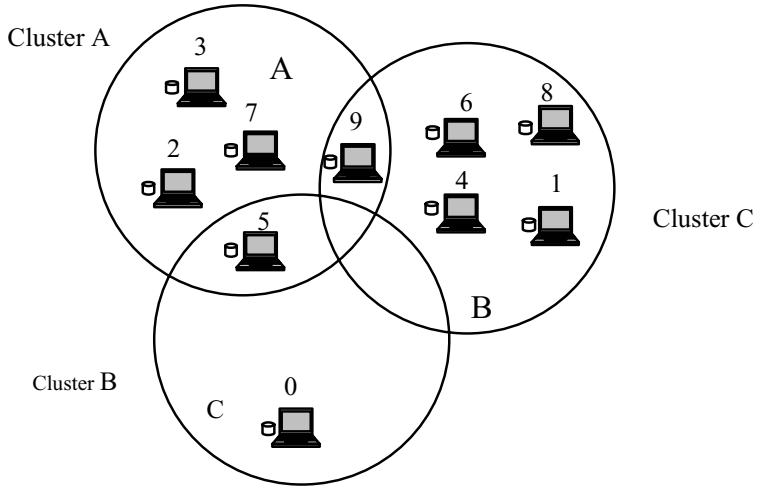

Fig. 1. The Clustering Architecture.

whether it is the data owner or if it has a fresh replica. If a node has the requested data, it will send a reply to the requester. If the node doesn't, it will simply forward the data request message to its neighbors. Nodes forward each data request and replica dissemination message generated by the same source only once.

4.2. Local update algorithm

An example of the clustering architecture is shown in Fig. 1. It consists of 10 mobile nodes in the network with the ID ranging from 0 to 9. Nodes 2, 3, 5, 7, and 9 can communicate with a maximum of two-hops via the cluster head. These nodes then form cluster A. Similarly, nodes 0 and 5 can form cluster B, and nodes 1, 4, 6, 8, and 9 join together to form cluster C. We designate nodes 5 and 9 as gateways, since they can communicate with nodes in more than one cluster. The local update algorithm ensures local data consistency. To illustrate the operation of this algorithm, we will consider two scenarios with reference to Fig. 1.

Scenario 1: Before a mobile node, say node 3, can perform an update operation, it sends an update lock request to the corresponding node, say node 2. If node 3 gets permission from node 2, node 3 then performs the update. If node 2 becomes unreachable after node 3 sent out its update, due to running out of power or disconnection from the network, node 3 will detect an update error, which indicates that node 3 cannot receive an ACK from node 2 before timing out. Instead of waiting passively or resending, node 3 will send the unsuccessful update to the cluster head (CH) for further processing. When a CH receives such an incomplete update, the CH performs the following actions: (i) stores the unsuccessful update D_{bj} in its undelivered update queue, (ii) sends D_{bj} to node 2, (iii) sends updated D_{bj} to CHs, and (iv) informs node 3.

Scenario 2: Another situation occurs when more than one node wants to update the same date item, a common scenario in distributed systems. For example, when a mobile node, say node 2, receives an update lock request for data item D_{bj} from node 3, node 2 first checks whether another node is updating D_{bj} by inspecting the flag bit L_{bj}, where 0 indicates that no one is updating and 1 indicates that someone is performing the update. If $L_{bj} = 0$, then node 2 sets the flag bit L_{bj} to 1 and sends a positive reply to the request, which means D_{bj} becomes updatable. If $L_{bj} = 1$, node 2 sends a negative ACK to the

request, which means that it is currently being updated by another node. Node 3 then waits for a random period of time. If node 3, which has locked the data item D_{bj}, left its current cluster or network before performing the update, the owner of the data item, node 2, can detect this disconnection based on the information maintained in its neighborhood table and reset the lock as $L_{bj} = 0$, thus avoiding a deadlock.

4.3. Global update algorithm

One of the main challenges in the replica control algorithm is maintaining data consistency across clusters. The global update algorithm is used to maintain global data consistency by ensuring that the data items stored in every cluster are consistent. The C-GRAP protocol proposed in [11] for heterogeneous grid databases handles multiple partitions using the concept of the contingency quorum. We believe that this protocol may be modified to be applied in MANETs. However, the modification should ensure that sufficient time is available to collect the read and write quorums to perform the right transaction in a dynamically changing network topology.

In our update algorithm, we assume that a data item can only be updated in its original cluster. Nodes located in other clusters are allowed only to read this data item and do not have the permission to modify it. Although, to some extent, this may cause write-unavailability problems for nodes in other clusters, this situation is suitable for some applications to maintain correctness. A MANET is an example of such applications in which data items require updates only by the nodes that created the data items or by the nodes located in the same partitions or clusters. Another advantage of doing this is that we can avoid global update conflict, which usually requires a relatively complex algorithm to control or ensure global data correctness across partitions in MANETs. In our scheme, when the original data owner receives an updated data item from one of the owner's neighbors, the original data owner will send the updated data to replica holders in other clusters and waits for the ACKs from them. If the replica sender does not receive an ACK from a certain replica holder within a certain time interval, the replica sender assumes that the replica holder is temporarily disconnected from the network. In order to successfully deliver the updated data later to that replica holder, the replica sender will send the data item to the CH, which will put the data item in the undelivered update queue. The updated data from the undelivered update queue will be delivered to the replica holder by invoking the distributed undelivered message queue algorithm to deliver the updated data.

5. Performance evaluation

5.1. Simulation environment

We simulated 120 mobile nodes using the NS-2 simulation tool [23]. To model node mobility, we used the Reference Point Group Mobility model (RPGM) model [16]. The RPGM model defines a logical reference point (center), which determines the group trajectory. This scheme captures both the independent and random movement behaviors of individual nodes. Two types of data are stored in each node, the node's own data and the replicas obtained from other nodes. Replicas are distributed to each cluster with the current number of clusters in the network determining the number of replicas generated by each node. Upon data update, each node sends the updated data to the replica holders to maintain data consistency. If no update of a replica is received within a certain time period, this replica becomes stale and will be deleted from that node. Replicas are updated periodically or at random intervals. Two different update strategies are studied: periodic and random. In a random update scheme, an update is

Table 1
Simulation Parameters

Parameters	Default values
Number of mobile nodes	120
Simulation area	1000 m × 1000 m
Transmission range	250 m
Transmission rate	5 Pkts/s
Traffic type	CBR
Speed (m/s)	0–25
Routing protocol	AODV
Length of simulation time	1000 s
Pause time	30 s
Number of simulations	20
Maximum packet size	1024 bytes
Update interval	10–100 s
Data Request interval	50–100 s

invoked every update period (T_u) seconds and T_u is randomly selected between 10 and 100 seconds. In a periodic update, the update is carried out constantly using a predefined interval. Most experiments were based on periodic updates. The simulation parameters are shown in Table 1.

5.2. Performance metrics

We considered four different metrics for performance evaluation: average data accessibility, packet delivery ratio, network traffic overhead, and replica distribution rate.

Average data accessibility: In our simulation, data accessibility is defined as the ratio of successful data requests to the total data requests made by a node. If this node receives a data item requested from other node(s) within a specified time threshold, the request is deemed successful. Otherwise, the request is considered a failure. Data accessibility was investigated as a function of the number of clusters, cluster head changes, transmission range, update intervals, and network size.

Average packet delivery ratio: Packet delivery ratio is defined as the ratio of the total number of packets received by the destination and the total number of packets sent by the source. The packet delivery was investigated as a function of the speed of the mobile node and the cluster size.

Average network traffic overhead: The average traffic overhead is defined as the total number of hops traveled for sending data request messages, getting data replicas, and distributing data replicas or updates. The network traffic was investigated as a function of the update interval.

Average replication distribution rate: The average replica distribution rate is defined as the ratio of the total number of replicas received by destination nodes to the total replicas sent to destinations. The replication distribution rate was investigated as a function of network size.

5.3. Discussion of simulation results

In this section, we present the results of the simulation experiments for each performance metric under various simulation parameter settings.

1. The effect of cluster size on packet delivery ratio: To evaluate transmission reliability, we used the average packet-delivery ratio as the performance metric. Figure 2 shows that the transmission reliability initially increased with the increase in cluster size and then declined beyond a cluster size of 15. This shows that a dense cluster has a higher packet-delivery ratio when compared to a sparse cluster. This occurs because a dense cluster has better connectivity and multiple forwarding nodes,

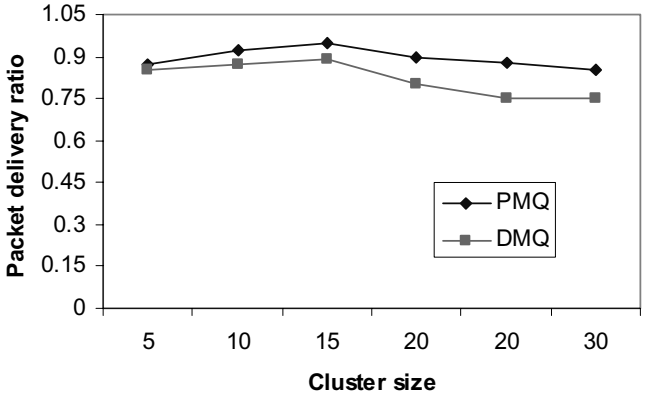

Fig. 2. Message Delivery Ratio and Cluster Size.

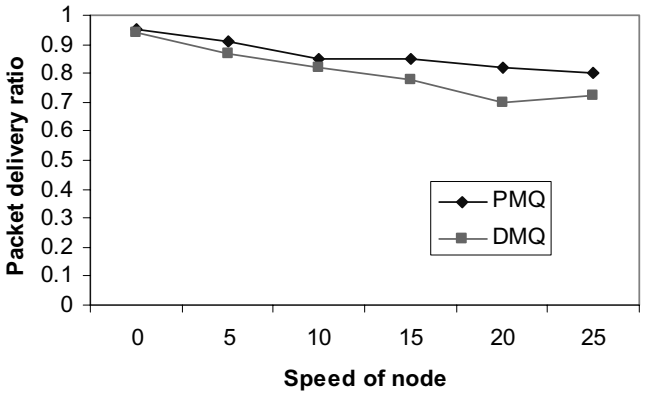

Fig. 3. Message Delivery Ratio and Moving Speed.

which result in higher message delivery. But too dense network results in congestion and result in decline. The simulation results indicate that the PMQ algorithm has a better delivery ratio than the DMQ algorithm. This happens because the prediction is more accurate at the lower cluster size; there is also less congestion. The message-delivery ratio slightly declined as the cluster size continued to rise for both algorithms. On the other hand, sparse clusters have lower connectivity and, hence, have relatively lower message-delivery ratios.

2. The effect of moving speed on packet delivery ratio: Figure 3 shows that the transmission reliability decreased with the increase in moving speed. Node mobility results in frequent topological changes since nodes may switch their cluster membership or the CH itself may change. The mobility of nodes can cause both link failures and link formation. This occurs because when nodes move away from each other a link fails, but when the nodes move toward each other or move together (even if the mobility is high), they can create new links or maintain an existing link. The simulation results indicate that for relatively low moving speeds, both schemes have similar packet-delivery ratios, with PMQ performing relatively better as the moving speed increases.

3. The effect of cluster head changes on data accessibility: Data accessibility was evaluated using the average data-accessibility ratio. We considered data accessibility with full cluster replication. Figure 4 shows that with the increase in CH changes, the data-accessibility ratio decreased. This occurs

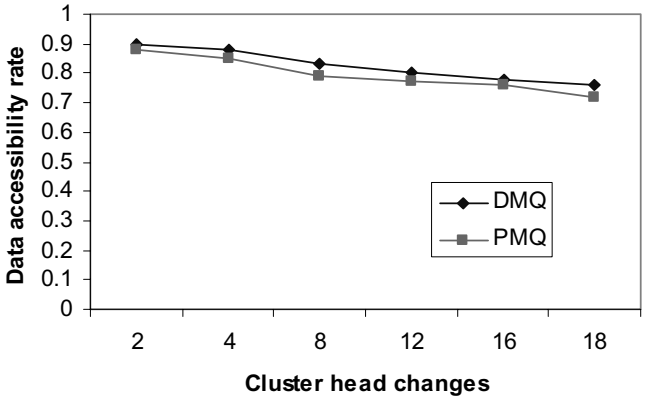

Fig. 4. Data Accessibility and Cluster Head Changes.

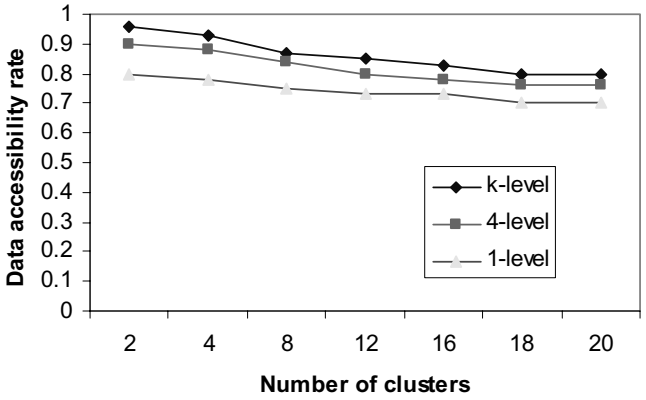

Fig. 5. Data Accessibility and Number of Clusters ($k = 6$).

because when the CH fails, a new CH has to be elected to take over the responsibility. However, since the CH election criteria ensure stability, the CH failure rate is relatively low and has little effect on performance.

4. The effect of number of clusters on data accessibility: Figure 5 shows that the data-accessibility ratio for DMQ. The results indicate that the data accessibility increased as the scope of replication increased. Broadcasting data to the farthest cluster results in better data access. Replicating among all CHs improved the accessibility to a larger degree. Thus, the simulation results indicate that as the replication scope of the cluster increases, performance becomes similar to that provided by a network-wide broadcast.

5. The effect of transmission range on data accessibility: Figure 6 shows that the data-accessibility ratio increased with the transmission range. With a larger transmission range, more nodes will be able to communicate directly and network partitioning occurs more rarely. Also, as the transmission range increased, the cluster size increased because more and more nodes were within the reach of each cluster. With smaller transmission ranges, however, nodes need more intermediate nodes in order to communicate with one another. This tends to reduce the data-accessibility ratio since these intermediate nodes may fail to relay messages. In general, the performance of both algorithms was similar, with DMQ performing relatively better.

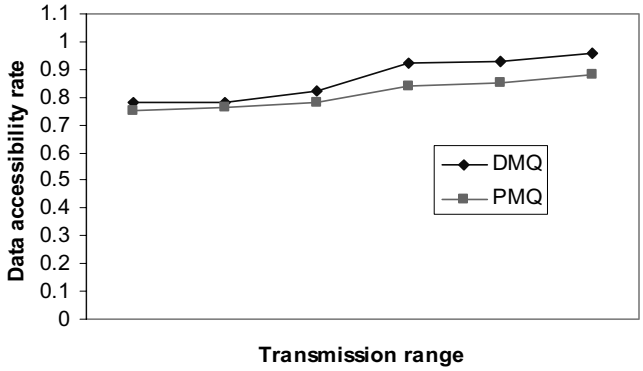

Fig. 6. Data Accessibility and Transmission Range.

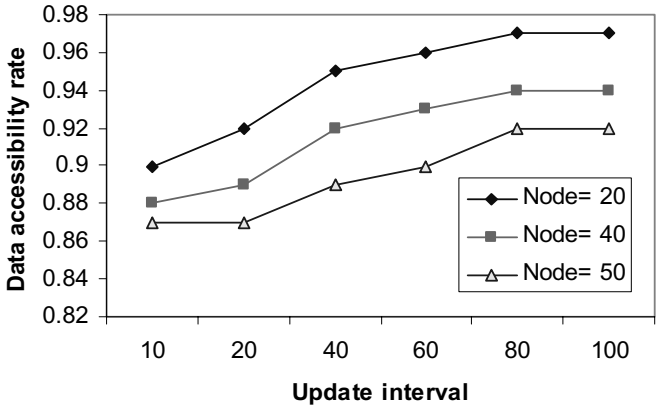

Fig. 7. Data Accessibility and Update Interval.

6. The effect of update interval on data accessibility: Figures 7 and 8 show the simulation results where the update periods of the data item were between 10 and 100 time units. In this experiment, scenarios consisting of networks of sizes 20, 40, and 50 were used to study performance. The results in Fig. 7 show that the average access rate in all scenarios was 92%. At all network sizes, when the update period became longer, the accessibility of each scenario also became higher, because the replicas are valid for a longer period. The disadvantage of a longer update interval is that the data items are not updated as timely as those in a shorter update period. In real implementation, the administrator chose the update period carefully to meet the requirement.

We also compared the data access rate of a random update with the access rate of a periodic update. While specific intervals were chosen between 10 and 100 time units for the periodic update, the random update was invoked at a random time between the same intervals. Figure 8 shows that the average access rate of the random update rose from 91% to 93%, while the average access rate for the periodic update rose from 92% to 95%.

7. The effect of network size on network traffic: Figure 9 shows that the amount of traffic decreased with the update intervals. Although the traffic of the replica distribution was reduced as the update period became longer, the traffic did not show a significant decrease when the update period became much longer.

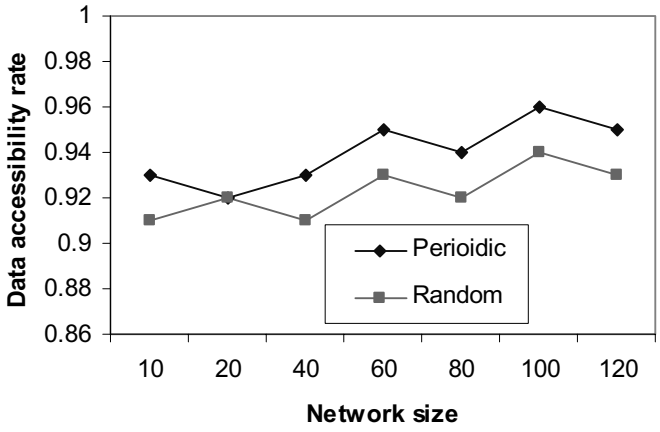

Fig. 8. Access Rate and Random Update.

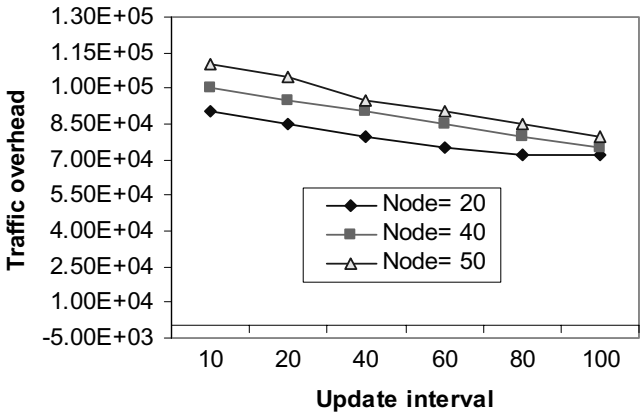

Fig. 9. Traffic Overhead and Update Period.

Figure 9 also shows that when the number of nodes exceeded 60, the traffic decreased slightly as the update interval increased, because as the network size became larger, the density of the nodes also grew. This implies that the average number of nodes in each cluster (cluster size) as well as the average number of neighbors of a node (nodal degree) increases. In turn, this will cause a traffic increase, since more nodes receive and forward data requests and data replies in such a cluster.

8. The effect of network size on the replica distribution rate: We investigated the replica distribution rate of our scheme. Figure 10 shows the simulation results for different update intervals as a function of network size. The results show that when the network size exceeded 60, the replica distribution slightly decreased. This was because when the network size increased, the density of nodes also increased, resulting in congestion and packet dropping.

Also, we noted from the graph that the replica distribution rate deceased when the network size increased beyond 60 nodes. This was because replica distribution bursts when the number of nodes in the network rises, generating more replicas. This results in the discarding of some replica packets in transit since the number of packets in the receiving queue may exceed the threshold, which was currently set to 50.

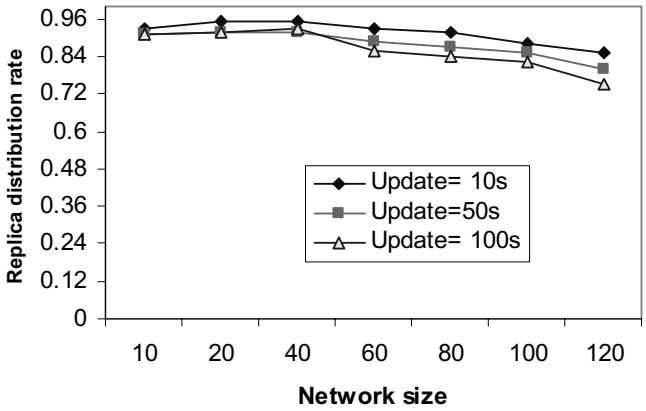

Fig. 10. Replica Distribution Rate and Network Size.

6. Conclusions and future work

In this paper, we proposed cluster-based replica-storage, dissemination, and update mechanisms for MANETs. Clustering architecture was used for data management. Replicas were distributed to each cluster, and updates were carried out periodically or at random time intervals. The clustering architecture improved the scalability of the proposed schemes and increased the reliability of data storage by replicating data at each cluster.

A discrete event simulation was carried out to evaluate the proposed replica storage, distribution, and mechanisms. The simulation experiments for data dissemination algorithms indicated that the DMQ algorithm guaranteed better data access while the PMQ algorithm achieved better message delivery in most scenarios. The simulation experiments on replica update strategies showed that the proposed schemes have higher replica distribution rate and data accessibility. The average data access rate of a periodic update was observed to be about 94% while that of a random update was 92%. The results generally suggest that when the update interval gets longer, the average data access rate increases. Thus, by choosing an appropriate update interval, the proposed scheme can increase data accessibility without affecting data consistency.

Our current study considered only replica management and update mechanisms to maintain data consistency in a frequently disconnected network environment. Further study is being undertaken on cache replacement strategies and cache sharing schemes among groups [6] to reduce the network traffic overhead. Investigating data consistency and security in the presence of multi-cluster replica updates is a topic of our future research.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. We would also like to acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for its support of this work.

References

[1] C.M. Bowman, P.B. Danzig and D. Hardy, The Harvest Information Discovery and Access System, *Computer Networks and ISDN Systems* **28**(1–2) (1995), 119–125.

[2] J. Byers, M. Luby and M. Mitzenmacher, Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed up Downloads, In Proc. of IEEE INFOCOM'99, 1999, pp. 275–83.

[3] G. Cao, L. Yin and C.R. Das, Cooperative Cache-Based Data Access in Ad Hoc Networks, *IEEE Computer* **37**(2) (2004), 32–39.

[4] S.B. Davidson, H. Garcia-Molina and D. Skeen, Consistency in Partitioned Networks, *ACM Computing Surveys* **17**(3) (Sept. 1985), 341—370.

[5] M.K. Denko, *Analysis of Clustering Algorithms in Mobile Ad Hoc Networks*, In Proc. of Int. Conf. on Wireless Networks, 2003, 98–105.

[6] M.K. Denko and J. Tian, Cooperative Data Caching and Prefetching in Wireless Ad Hoc Networks, In Proc. of 2nd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WIMOB 2006), 2006, 38–44.

[7] R. Friedman, M. Gradinariu and G. Simon, Locating Cache Proxies in MANETs, 5th ACM MOBIHOC, May 2004, Tokyo, Japan.

[8] F. Garcia, J. Solano and I. Stojmenovic, Connectivity based k-hop clustering in wireless networks, *Telecommunication Systems* **22**(1–4) (2003), 205–220.

[9] M. Gerla and J. Tsai, Multicluster, Mobile, Multimedia Radio Network, *in Wireless Networks* **1**(3) (1995).

[10] S.K. Goel, M. Singh, D. Xu and B. Li, *Efficient Peer-to-Peer Data Dissemination in Mobile Ad Hoc Networks*, Vancouver, B.C., Canada, 2002.

[11] S. Goel, H. Sharda and D. Taniar, Replica synchronization in grid databases, *In International Journal of Web and Grid Services* **1**(1) (2005), 87–112.

[12] T. Hara, K. Harumoto, M. Tsukamoto and S. Nishio, Dynamic replica Allocation Using Database Migration in Broadband Networks, In Proc. of IEEE ICDCS 2000, 2000, 376–384.

[13] T. Hara, Effective Replica Allocation in ad hoc Networks for Improving Data Accessibility, In Proc. IEEE INFOCOM 2001, 2001, 1568–1576.

[14] T. Hara, Replica Allocation in Ad Hoc Networks with Periodic Data Update, In Proc. of the Third International Conference on Mobile Data Management (MDM. 2002), 79–86.

[15] T. Hara, Replica Allocation in Ad Hoc Networks with Data Update, *In ACM MONET* **8**(4) (2003), 343–354.

[16] X. Hong, M. Gerla, G. Pei and C.C. Chiang, A Group Mobility Model for Ad Hoc Wireless Networks, In Proc. ACM/IEEE MSWiM'99, Seattle, WA, 1999, 53–60.

[17] S. Ishihara, M. Tamor, T. Mizuno and T. Watanabe, *Replication of Data Associated with Locations in ad hoc Networks*, In Proc. of the 2004 IEEE Int. Conf. on Mobile Data Management (MDM'04).

[18] G. Karumanchi, S. Muralidharan and R. Prakash, Information Dissemination in Partitionable Mobile Ad Hoc Networks, In Proc. of 18th IEEE Symposium on Reliable Distributed Systems, Lausanne, Switzerland, 1999, 4–13.

[19] Q. Li and D. Rus, Sending Messages to Mobile Users in Disconnected Ad Hoc Wireless Networking, In Proc. of ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MOBICOM), 2000.

[20] H. Lu and M. Denko, Reliable Data Storage and Dissemination in Mobile Ad hoc Network, In Proc. of International workshop on Theoretical and Algorithmic Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks, June 2004, Chicago, USA.

[21] H. Lu and M.K. Denko, Replica Update Strategies in Mobile Ad Hoc Networks, In Proc. of 2nd IEEE/IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005), Dubai, UAE, 2005, 302–306.

[22] J. Luo, J.-P. Hubaux and P.T. Eugster, PAN: Providing Reliable Storage in Mobile Ad Hoc Networks with Probabilistic Quorum Systems, In Proc. of MOBIHOC'03, Annapolis, Maryland, 2003, 1–12.

[23] S. McCanne and S. Floyd, *Network Simulator*, http://www.isi.edu/nsnam/ns/.

[24] F. Nimbona and S. Pierre, Stability and Resource Consideration Algorithm for Clustering in Mobile Ad Hoc Networks, In Proc. of Int. Conf. on Wireless Networks, 2004.

[25] C. Okino and G. Cybenko, Modeling and Analysis of Active Message in Volatile Networks, In Proc. of 37th Allerton Conf. on Communication, Control, Computing, Monticello, IL, 1999.

[26] F. Sailhan and V. Issarny, Cooperative Caching in Ad Hoc Networks, In proc. of the 4th Int. conf. on Mobile Data Management, Melbourne, Australia, Jan 2003.

[27] R. Shah and N.C. Hutchinson, Delivering Messages in Disconnected Mobile Ad Hoc Networks, In Proc. of ADHOC-NOW 2003,LNCS 2865, 72–83, 2003.

Mieso K. Denko received a BSc in Statistics and Mathematics from the Addis Ababa University in 1988. He received his MSc in 1994 from the University of Wales, UK and his PhD in 2002 from the University of Natal, South Africa both in Computer

Science. Dr. Denko's research interests are in the broad areas of wireless networks, mobile ad hoc networks, mobile & pervasive computing, ad hoc sensor networks and wireless mesh networks. He has published a number of articles that have appeared in international journals, conferences and workshops. He is currently co-editing a forthcoming book on mobile ad hoc and pervasive communications. Dr. Denko is currently with the Department of Computing and Information Science, University of Guelph, Canada.

Lu Hua received a B.E in Computer Engineering from Hangzhou University of Commerce, Hangzhou, China in 1999. He received his MSc in Computer Science from University of Guelph, Guelph, Canada, in 2005. His research interests include mobile routing protocols, mobility management in wireless networks, mobile sensor networks and data management in mobile ad hoc networks.