MANAGING DATA REPLICATION IN MOBILE AD-HOC NETWORK DATABASES (Invited Paper)[*]

Prasanna Padmanabhan
School of Computer Science
The University of Oklahoma
Norman OK, USA
prasannap@yahoo-inc.com

Dr. Le Gruenwald
School of Computer Science
The University of Oklahoma
Norman, OK, USA
ggruenwald@ou.edu

Abstract — **A Mobile Ad-hoc Network (MANET) is a collection of wireless autonomous nodes without any fixed backbone infrastructure. All the nodes in MANET are mobile and power restricted and thus, disconnection and network partitioning occur frequently. In addition, many MANET database transactions have time constraints. In this paper, a Data REplication technique for real-time Ad-hoc Mobile databases (DREAM) is proposed that addresses all those issues. It improves data accessibility while considering the issue of energy limitation by replicating hot data items at servers that have higher remaining power. It addresses disconnection and network partitioning by introducing new data and transaction types and by considering the stability of wireless link. It handles the real-time transaction issue by replicating data items that are accessed frequently by firm transactions before those accessed frequently by soft transactions. DREAM is prototyped on laptops and PDAs and compared with two existing replication techniques using a military database application. The results show that DREAM performs the best in terms of percentage of successfully executed transactions, servers' and clients' energy consumption, and balance of energy consumption distribution among servers.**

I. INTRODUCTION

MANET is a collection of wireless autonomous nodes that may move unpredictably, forming a temporary network without any fixed backbone infrastructure [4]. All the nodes in MANET are mobile and, thus, restricted by power. These nodes not only can communicate with nodes that are within their communication ranges, but also can communicate with nodes that are outside their transmission ranges using multi hop communication.

Since no fixed infrastructure is required, they fit well in military, rescue operations and sensor networks [8]. Moreover many applications in this environment are time-critical and, hence, their transactions should be executed not only correctly, but also within their deadlines.

In a distributed database system, data are often replicated to improve reliability and availability. However one important issue to be considered while replicating data is the correctness of the replicated data. Since nodes in MANET are mobile and have limited energy, disconnection may occur frequently, causing a lot of network partitioning. Data that is available in mobile hosts in one partition cannot be accessed by mobile hosts in another partition. The issues related to data replication in MANET databases are as follows:

- *Server Power Consumption:* Servers in MANET run on battery power. If a server has low power remaining and if it is replicated with many frequently accessed data items, then the frequent data access requests for these hot data will drain its power.
- *Server mobility:* Due to their mobility, servers might sometimes move to a place where they could not be reached by any other server or client.
- *Client mobility:* Clients that query servers for information are also mobile. Clients usually send their transactions to the nearest servers to get a quick response. The decision to replicate data items at a particular server may be based on the access frequencies of data items from that specific server. Hence, the decision to replicate data items at appropriate servers must be dynamic and based on the current network topology.
- *Client Power:* Clients in MANET also run on battery power. If a client waits too long for its transactions' results, it might lose its power rapidly. The replication technique should be able to replicate data

[*] This work was supported in part (while serving at) by the National Science Foundation (NSF) and NSF Grant No. IIS-0312746

items at appropriate servers in such a way that a client might be able to access its requested data items from its nearest server which has the least workload.

- *Time-critical applications*: Many MANET applications, like rescue networks, military operations, are time-critical. Data replication should be used to improve data accessibility and performance of the system, thereby reducing the time to execute transactions. Transactions with short deadlines should be sent to the nearest servers that have the least workload for their execution. These transactions should also be executed before other transactions that have longer deadlines.
- *Frequent Disconnection and Network Partitioning*: Disconnection and network partitioning is a severe problem in MANET as servers in one partition that hold required data items cannot provide services to other clients and servers in different partitions.
- *Update transactions*: If there are no constraints in storage space and if there are read-only transactions, data accessibility and performance of MANET databases can be increased by fully replicating the database in all the servers. However, maintaining consistency in a fully replicated database is a major issue when there are frequent update transactions.

No single existing replication technique considers all of the above issues. In the poster paper [17b], we have briefly presented a replication technique, called DREAM that addresses all the above issues. In this paper, we discuss this technique and its performance evaluation in detail. The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the underlying MANET database system architecture. Sections 4, 5 and 6 present DREAM, the prototype model and the performance evaluation results, respectively. Finally, Section 7 concludes the paper with future work.

II. RELATED WORKS

A data replication technique that replicates data items based on their access frequencies and the current network topology is proposed in [6]. Hot data are replicated before cold data items. If the access characteristics of data items are similar, there could be replica duplications at many mobile nodes. Hence, two other techniques to reduce replica duplication between mobile nodes are proposed in [6]. They also detect network partitioning and replicate hot data items before such a partitioning occurs to improve data accessibility. However in those techniques, when there is a replica duplication between any two connected mobile nodes, one of the duplicate replicas is replaced by another hot data item, irrespective of how high the access frequency of the replaced data item is or how low the access frequency of the new data item is.

A data replication technique that replicates data items at multiple nodes and employs quorum based strategies to update and query information is proposed in [11]. It sends the update information to nodes in such a way that other nodes while querying for this update information gets the most updated information, and thereby, reducing inconsistency and dirty read transactions. It determines the time to send the updates, where to send them and which nodes to query for the information in such a way that it mitigates the impacts of network partitioning. Another way of disseminating the update information to mobile nodes is by data broadcasting as proposed in [19]. This replication technique delays the updates to replicas for performance and bandwidth considerations. It also determines what updates to broadcast and when to broadcast them in an efficient manner.

A set of protocols that use a gossip-based multicast protocol to probabilistically disseminate updates in a quorum system is proposed in [15]. It achieves high reliability even when there are large concurrent update and query transactions. A new metric for evaluating wireless link robustness that is used to detect network partitioning is proposed in [7]. Its decision to replicate data items is taken not only at the time of detecting network partitioning, but also during the time when the wireless connections become bad in terms of reliability, bandwidth and delay.

Roam [18] is a replication technique that attempts to provide data availability to mobile hosts irrespective of the mobility of the hosts. It models the mobility of hosts by grouping them into wards and determines periods of motion of the mobile hosts. Ward masters are elected to provide communication across wards, but hosts belonging to the same ward may directly communicate with each other. Roam maintains consistency of replicas across the network, irrespective of the locations of movements' different hosts. None of the above replication techniques addresses the issues related to real-time database transactions and mobile hosts' power limitation. It should also be noted that network partitioning might occur not only due to mobile hosts' mobility, but also due to battery power drainage of some mobile hosts.

III. DECENTRALIZED MANET DATABASE ARCHITECTURE

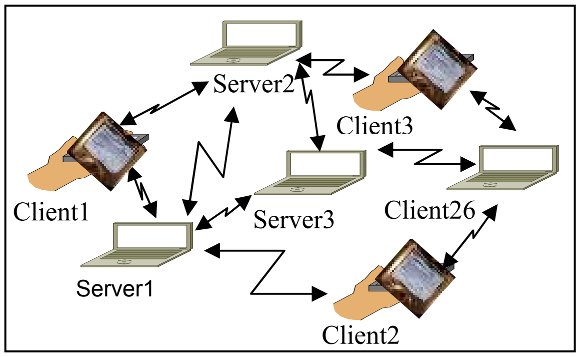


Figure 1: Decentralized MANET Database Architecture

Depending on communication strength, computing power, disk and memory storage size, mobile hosts can be classified into two groups: Clients and Servers. Clients are equipped with reduced memory and computing capabilities. They store only the Query Processing module of the Database Management System (DBMS) that allows them to submit transactions to appropriate servers and receive results. Servers are equipped with higher memory and computing capabilities; they store the complete DBMS. For example, soldiers in battlefields, who are equipped with handy portable computers like PDAs and smart phones, could be considered as clients, while tanks and humvees equipped with high end portable computers like laptops could be considered as servers.

This paper assumes a decentralized architecture, where clients are free to communicate (single-hop or multi-hop) and submit their transactions to any of the available servers in the network as shown in Figure 1. This architecture does not place reliance on any centralized server and, thus, improves system resilience by avoiding a single point of failure.

IV. PROPOSED MANET DATA REPLICATION TECHNIQUE: DREAM

A data replication for MANET databases, called DREAM, is proposed in this section. It extends the techniques proposed in [6] that have briefly been reviewed in Section 2 to consider additional issues including mobile hosts' power limitation, real-time database transactions as well as various data and transaction types that exist in many MANET applications. DREAM improves data accessibility while addressing the issue of power limitation by replicating hot data items before cold data items at servers that have high remaining power. It handles the real-time transaction issue by giving a higher priority for replicating data items that are accessed frequently by firm transactions than those accessed frequently by soft transactions It addresses disconnection and network partitioning by introducing new data and transaction types and by determining the stability of wireless links connecting servers. The remaining energy of connecting servers is also used to measure their link stability. The data and transaction types are proposed in Sections A and B, respectively, after gathering the database requirements of a fire rescue and a battlefield application from the Norman Fire Department and the Reserve Officers Training Corps (ROTC) at the University of Oklahoma. Each server in DREAM that holds the original copy of a data item is termed its primary copy server and the other servers that hold the replicas of the data item are termed its secondary copy servers. DREAM addresses the replica synchronization issue by maintaining two timestamps that indicate when a particular data item is updated in its primary and secondary copy servers.

DREAM is composed of three main parts. The first part determines the data items to be replicated and the servers in which they have to be replicated. The second part determines how the allocated replicas can be accessed for transaction processing based on their data and transaction types. The third part identifies the way to synchronize the replicas. These parts are discussed in subsequent sections.

A. Data Types

DREAM's data types are shown in Figure 2. Data items are classified into read-only and read-write data items. Read-write data items can be further classified into two types: Temporal and Persistent. The former are those data items the values of which are valid only for a certain period of time. The location of a soldier and the location of an enemy are examples of temporal data items as they move frequently in a battlefield. However, persistent data items are valid throughout their existence in the database. Locations of medical assistance and a list of wounded soldiers are examples of persistent data items. All read-write data items can be further classified into periodic and aperiodic update data items. Periodic update data items are those data items that are updated periodically at fixed intervals of time. For example, the location of a soldier may be updated frequently at a constant interval of time. Aperiodic-update data items are those data items that are updated at random intervals of time. For example, the location and information about an enemy may be updated at random time intervals.

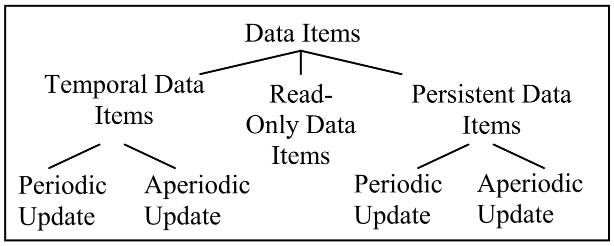


Figure 2: Data Types

B. Transaction Types

The transaction types proposed in DREAM are shown in Figure 3. Transactions, both firm and soft, are classified as read and write transactions.

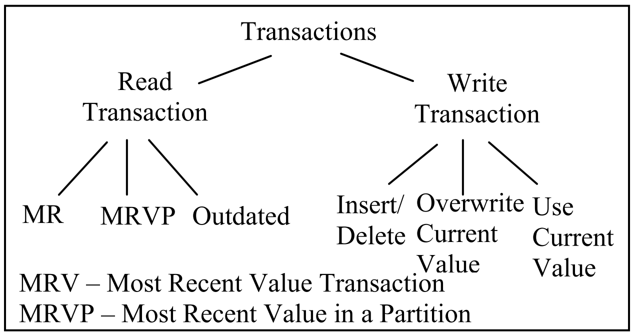


Figure 3: Transaction Types

Read transactions: they are further classified into three sub-categories depending on the freshness of the data read. Some transactions need the most recent values of data items across all the database servers in the network for transaction processing. These transactions are called Most Recent Value (MRV) transactions. For example, a military officer might need the most recent location of an enemy to launch a missile. Some other transactions give less importance to the freshness of the data for transaction processing and can be executed even if the retrieved data is stale. Such transactions are called outdated transactions (OD). For example, a soldier might need the weather information of a particular place to prepare for his or her clothing.

Another type of read transactions is called the Most Recent Value in a Partition (MRVP) transactions. These transactions require the most recent values of data items across all the database servers in a network partition. The result of a MRVP transaction may or may not be the most recent value of the requested data item across the entire database. For example, a wounded soldier might query for the location of medical assistance. This transaction would have ideally been a MRV transaction. However in a MANET environment, not all nodes are connected all the times and, hence, the most recent values of data items across all the database servers may not sometimes be determined. An outdated transaction for this case is also not an ideal one. Therefore, the most recent values of the required data items across all the database servers in the network partition in which the wounded soldier currently resides should be returned back to him or her.

Write transactions: they are further divided into three sub-categories depending on how their data updates occur. Some write transactions insert or delete records into the database. These transactions are called Insert/Delete transactions. Another type of write transactions is called the Use Current Value (UCV) transactions. These transactions need the current values of data items for transaction processing. For example in a military application, a soldier might update the number of wounded soldiers by a certain value. For this transaction to be successfully executed, the number of currently wounded soldiers must be known.

Another type of write transactions is called the Overwrite Current Value (OCV) transactions. These transactions overwrite the current values of data items and can be executed successfully irrespective of the current values of the data items. For example, soldiers in a battlefield may update their locations periodically and such update operations on their locations overwrite the current values of their locations.

C. What and Where to Replicate?

Figure 4 shows this part of DREAM. Each server in DREAM can store only a certain number of data items, called maximum capacity. The first step of the algorithm is to calculate the weighted access frequencies of data items based on their data and transaction types. After determining their weighted access frequencies, data items with higher weighted access frequencies (hot data items) are replicated before data items with lower weighted access frequencies (cold data items) in servers that have the maximum remaining power. After storing the hot data items at appropriate servers, if there are any redundant data items among neighboring servers, they are eliminated depending upon the stability of the links connecting them and the access frequencies of the next available hot data items. Such a decision to replicate data items at appropriate servers is taken every time during a certain period of time called the *relocation period* [6].

1. Computing Access Frequencies based on Data and Transaction Types

Access frequency of a data item d at a particular server s, $\text{Access\_Frequency}_d^s$, is the number of times that d is accessed at s. From the access logs, similar to [6], the access frequency of each data item at each server is computed. In addition, in DREAM, the numbers of times a data item is accessed by Firm, Soft, MRV and Non-UCV transactions at all servers are computed. These are computed only once by all the servers when running the replication algorithm for the first time. These access frequencies are then further calculated using a weight factor based on the data and transaction types presented in Sections 4.1 and 4.2. The resulting access frequencies are thus the weighted ones to reflect replication prioritization as shown in the following sections.

a) Firm and Soft Transactions: if a data item is accessed more often by firm transactions than another data item is, then the former data item is given a higher priority to be replicated than the latter. This is to reduce the number of transaction aborts since firm transactions must be aborted if they missed their deadlines. Firm transactions' execution time can decrease considerably if their required data items reside in the servers to which the requests were sent. The replication priority is set by assigning weighted access frequencies to data items that are accessed by firm transactions using the below formula where $\text{Access\_Frequency}_{d\_Firm}^s$ is the number of times data item d is accessed by firm transactions at server s.

$$Access\_Frequency_d^s = Access\_Frequency_d^s + Access\_Frequency_{d\_Firm}^s * (Access\_Frequency_{d\_Firm}^s / Access\_Frequency_d^s)$$

b) Temporal Data Item: a temporal data item is valid only for a certain time period called its *age*. If the time remaining during which a temporal data item is valid is greater than the relocation period, then the probability for successfully accessing that temporal data item until the next relocation period is high as the temporal data item is valid throughout the entire relocation period. However, if the remaining valid time interval is less than the relocation period, then the temporal data item is valid for only some portion of the relocation period. The ratio between the remaining valid time interval and the relocation period is called the *Age Relocation Ratio* of a temporal data item. A data item with a higher age relocation ratio is replicated before the one with a lower age relocation ratio. This is because the probability of successfully executing a transaction that accesses the former is more than the one accessing the latter as the former is valid for a longer interval of time. Hence, the weighted access frequency of a temporal data item is calculated based on its age relocation ratio using the following formula where $\text{Age\_Relocation\_Ratio}_d^{\,s}$ is the *Age Relocation Ratio* of temporal data item *d* at server *s*.

$$Access\_Frequency_d^s = Access\_Frequency_d^s * Age\_Relocation\_Ratio_d^s$$

c) Read-Write Data Item: a read-write data item that is accessed frequently by UCV transactions is given a lower priority to be replicated than a read-write data item, which has the same access frequency as the former and is accessed frequently by Non-UCV transactions. This is due to the fact that the probability of executing an UCV transaction is lower than the probability of executing a Non-UCV transaction as the latter can be executed irrespective of the current values of their required data items. Similarly, the probability of successfully executing a MRV transaction is lower than the probability of successfully executing a Non-MRV transaction. Thus, a read-write data item that is accessed frequently by MRV transactions is given a lower priority to be replicated than a read-write data item which has the same access frequency as the former and is accessed frequently by Non-MRV transactions.

2. Replica Allocation and Redundancy Elimination

By comparing the movements of two connected neighboring servers, the distance between them can be calculated. Based on their distance, transmission ranges and velocity, the time in which they would be disconnected can be estimated as presented in [6]. This time is called the disconnection time of these two servers. If the disconnection time of two servers is greater than the relocation time period, these two servers can share data reliably until the next relocation time period. However, unlike [6] which does not address mobile hosts' power limitation, we believe that even if two servers are within each others' transmission ranges, they might not be able to communicate with each other if they are out of power. Hence, in DREAM, the link stability connecting two servers until the next relocation period is given by the formula:

*Reliability Ratio = Percentage of Server Power Remaining * (disconnection time / relocation period)*

A higher reliability ratio between two servers means that the link connecting them is more stable to share data between them. For example, if the reliability ratio of two servers is 0.5, then only for 50% of the relocation period can these servers share data, while for the rest of the time they might be disconnected.

After determining the access frequencies and the reliability ratio of all servers, data items in the descending order of their access frequencies are assigned to servers until the max capacity of data items in those servers has been reached. If the access frequencies of data items are similar at many servers, the same data items would be replicated at those servers [6]. Replica duplication among neighboring servers can be eliminated by replacing a duplicate data item with the next highest accessed data item as presented in [6]. However, if those neighboring servers get disconnected in the future, data accessibility would even decrease because of such a replacement. Hence, in DREAM, the decision to eliminate redundancy is taken only if it improves data accessibility. Assume there is a replica duplication of a data item d_x between two neighboring servers, s_i and s_j. The decision to remove this redundancy in DREAM is based on the following two conditions:

a) *One of the servers (say s_i) is the primary copy server of d_x:* in this case, the next highest accessed data item in s_j, d_y, is computed. This data item d_y can replace d_x in s_j only if the link connecting s_i and s_j is stable so that all requests for d_x in s_j can be successfully executed by forwarding the request to s_i. There is, however, no use in replacing d_x by d_y if the difference between the access frequencies of d_x and d_y in s_j is very high or if the link connecting s_i and s_j is unreliable. As discussed above, the reliability ratio of two servers indicates the stability of the links connecting them. It is beneficial to replace d_x by d_y in s_j only if the sum of the number of times that d_y can be accessed from s_j ($\text{Access\_Frequency}_{dy}^{sj}$) and the number of times that d_x can be accessed from s_i ($\text{Reliability\_Ratio}_{si}^{sj}$ * $\text{Access\_Frequency}_{dx}^{sj}$) is greater than the number of times that d_x can be accessed from s_j ($\text{Access\_Frequency}_{dx}^{sj}$).

b) Both of the servers are secondary copy servers of d_x: when these servers, s_i and s_j, hold only the replica and not the original copy of d_x, the decision to remove this redundancy is based on the access frequencies of the next frequently accessed data items in both s_i and s_j. The next frequently accessed data items, d_u and d_v, for s_i and s_j, respectively, are computed. As in case (a), DREAM first determines if it is beneficial to replace d_x by d_u in s_i, and d_x by d_v in s_j. If either of one of them is beneficial, then d_x is replaced by the appropriate data item at the server in which it is beneficial. If none of them is beneficial, then the redundancy is not eliminated. If both of them are beneficial, the redundancy is eliminated in the server in which more benefit is obtained.

```
What_to_Replicate_in_Which_Server(data_item d, server s, data_type, Access_Frequency$_d^s$, Relocation_Period)
/* Valid_Remaining_Time$_d^s$ – the time remaining until which
the temporal data item d in the server s is valid;
Absolute_Validity_Interval$_d^s$ – the absolute validity interval of
data item d in server s;
Timestamp_Current_Value$_d^s$ – the time when the current value
of the temporal data item d is obtained in server s */
Begin
  /* Compute the access frequencies of data items based on
     their data types */
  If (d is a Temporal Data Item)
    Compute Valid_Remaining_Time$_d^s$ using the formula
     Valid_Remaining_Time$_d^s$ = Timestamp_Current_Value$_d^s$ +
          Absolute_Valid_Interval$_d^s$ – $T_{now}$
    If (Valid_Remaining_Time$_d^s$ >= Relocation_Period)
      Age_Relocation_Ratio$_d^s$ = 1
    Else
      Age_Relocation_Ratio$_d^s$ = Valid_Remaining_Time$_d^s$ /
                                   Relocation_Period
    End If
    /* Access frequencies of temporal data items are computed
       based on their remaining valid time period */
    Access_Frequency$_d^s$ = Access_Frequency$_d^s$ *
                             Age_Relocation_Ratio$_d^s$
  End If
  If (d is a read-write data item)
    If (s is not the primary copy server of d)
      /* assume p is the primary copy server of d
         UCV and MRV transactions accessing d should be
         forwarded to p */
      Access_Frequency$_d^s$ =Access_Frequency$_{d\_Non\ UCV}^s$ +
        Access_Frequency$_{d\_Non\ MRV}^s$ + Reliability_Ratio$_p^s$ *
        (Access_Frequency$_{d\_UCV}^s$ + Access_Frequency$_{d\_MRV}^s$)
    End If
  End If
    Store maxCapacity number of data items in descending
    order of their access frequencies in server s
    For each server $s_i$
      For each server $s_j$ adjacent to $s_i$
        For all data items $d_x$ that is redundant
            between $s_i$ and $s_j$
          If (one is a primary copy server (say $s_i$) and the
              other a secondary copy (say $s_j$))
            Find the next highest access frequency data item,
              $d_y$, in $s_j$
              /* Replace $d_x$ by $d_y$ in $s_j$ only if the sum of the
              number of times that $d_y$ can be accessed from $s_j$
              (Access_Frequency$_{dy}^{sj}$) and the number of
              times that $d_x$ can be accessed from $s_i$
              (Reliability_Ratio$_{si}^{sj}$ * Access_Frequency$_{dx}^{sj}$) is
              greater than the number of times that $d_x$ can be
              accessed from s Access_Frequency$_{dx}^{sj}$); $d_x$
              should be replaced by $d_y$ in $s_j$ only if it would
              increase data accessibility */
              If (Access_Frequency$_{dy}^{sj}$ + Reliability_Ratio$_{si}^{sj}$
                  * Access_Frequency$_{dx}^{sj}$ >
                              Access_Frequency$_{dx}^{sj}$)
                Replace $d_x$ by $d_y$ in $s_j$
              End If
          Else if (Both $s_i$ and $s_j$ are secondary copy servers)
            /* either change one or change none */
            Find the next highest access frequency data
              items $d_u$ and $d_v$ for $s_i$ and $s_j$, respectively.
            /* Can_$d_u$_Replace_$d_x$ – Check if the data
            accessibility would be improved if $d_x$ is
            replaced by $d_u$ in $s_i$. This is done by checking if
            the sum of the number of times that $d_u$ can be
            accessed from $s_i$ (Access_Frequency$_{du}^{si}$) and
            the number of times that $d_x$ can be accessed
            from $s_j$ (Reliability_Ratio$_{sj}^{si}$ *
            Access_Frequency$_{dx}^{si}$) is greater than the
            number of times that $d_x$ can be accessed from $s_i$
            (Access_Frequency$_{dx}^{si}$) */
            Can_$d_u$_Replace_$d_x$ = Access_Frequency$_{du}^{si}$ +
                Reliability_Ratio$_{sj}^{si}$ * Access_Frequency$_{dx}^{si}$
                  > Access_Frequency$_{dx}^{si}$
            /* Can_$d_v$_Replace_$d_x$ – Check if the data
            accessibility would be improved if $d_x$ is
            replaced by $d_v$ in $s_j$. */
            Can_$d_v$_Replace_$d_x$ = Access_Frequency$_{dv}^{sj}$ +
                Reliability_Ratio$_{si}^{sj}$ * Access_Frequency$_{dx}^{sj}$
                  >  Access_Frequency$_{dx}^{sj}$
            If (Not Can_$d_u$_Replace_$d_x$ &&
                Not Can_$d_v$_Replace_$d_x$)
              The redundancy is not eliminated in
                  either of the servers
            Else If (Can_$d_u$_Replace_$d_x$ &&
                     Can_$d_v$_Replace_$d_x$)
              /* If the data accessibility is more when $d_x$
              is replaced by $d_u$ in $s_i$ than when $d_x$ is
              replaced by $d_v$ in $s_j$, then $d_x$ will be
              replaced by $d_u$ in $s_i$; else $d_x$ will be replaced
              by $d_v$ in $s_j$ */

              If (Access_Frequency$_{dv}^{si}$+ Reliability_Ratio$_{sj}^{si}$ *
                  Access_Frequency$_{dx}^{si}$ > Access_Frequency$_{dv}^{sj}$ +
                  Reliability_Ratio$_{si}^{sj}$ * Access_Frequency$_{dx}^{sj}$)
                Replace $d_x$ by $d_u$ in $s_i$
              Else
                Replace $d_x$ by $d_v$ in $s_j$
              End If
            Else If (Can_$d_u$_Replace_$d_x$)
              Replace $d_x$ by $d_u$ in $s_i$
            Else
              Replace $d_x$ by $d_v$ in $s_j$
            End If
        End For
      End For
    End For
End What_to_Replicate_in_Which_Server
```

Figure 4: What and Where to Replicate in DREAM

D. How to Access Replicas?

Due to space limitation, we present here only the basic ideas of the algorithm; interested readers are referred to [17] for the detailed algorithm.

Once data items are replicated at appropriate servers, they are accessed in different ways based on the proposed data and transaction types. When there is a request for a data item *d* to a server that is the primary copy server of *d*, *d* can be accessed directly from that server, irrespective of its data type. If the initiated transaction is a write transaction, the primary copy server, after updating its original copy of *d,* broadcasts its update timestamp to indicate to the other secondary copy (replica) servers that hold *d* that their replicas are out of synchronization.

In contrast, if the coordinating server is not the primary copy server of the requested data item, the data item is accessed in different ways based on the data and transaction types as discussed in the following sections. Every server *s* that holds a replica of data item *d* has two timestamps: the time when *d* is updated in *s,* $\text{Local\_Update\_Timestamp}_d^s$, and the time when *d* is updated at its primary copy server, $\text{Primary\_Update\_Timestamp}_d^s$. These two timestamps are used to determine if the replica is in synchronization with the original copy.

1. Read Transactions

If the requested data item is a read-only data item, it can be accessed from any server that holds it. Similarly, if the initiated transaction is an OD transaction, it can be accessed from any server that holds it. For both of these two cases, if the requested data item is available at more than one server, the decision to choose an appropriate server is based on the real time transaction type. A firm transaction is sent to the nearest server with the least workload, while a soft transaction is sent to the highest energy server with the least workload, for transaction processing. The objective is to reduce the number of transaction aborts and, at the same time, balance the energy consumption distribution among servers.

If the requested transaction is a MRV transaction, the requested data item should be accessed from the server that has its most recent value among all the servers that hold it. If the requested transaction is a MRVP transaction, the requested data item should be accessed from the server that has its most recent value among all the servers in its network partition that hold it. Based on the $\text{Local\_Update\_Timestamp}_d^s$ of all servers *s*, the most recent value of data item *d* can be determined.

A periodic update data item is updated once every certain period of time called its update frequency. Hence, a MRV or a MRVP transaction accessing a periodic update data item can access it from any server that has updated it during its last known update time interval. For example, consider a periodic data item *d* that is updated every one hour. A server *s* has the most recent value of *d* if the difference between the current time (T_{now}) and the last update timestamp of *d* in *s* ($\text{Local\_Update\_Timestamp}_d^s$) is less than its update frequency ($\text{Frequency\_Update}_d^s$), which is one hour in this example.

2. Write Transactions

If the transaction is an update transaction and if the coordinating server is connected to the primary copy server of the requested data item, the update transaction is forwarded to the primary copy server for transaction processing. If the coordinating server holds a replica of the requested data item, and if the transaction is not an UCV transaction, the local replica is also updated as further read requests for that data item can be accessed directly from the local replica.

However, if the coordinating server is not connected to the primary copy server, and if the transaction is not an UCV transaction, the update transaction is forwarded to the server that holds the requested data item. If there is more than one server that holds the requested data item, a firm transaction is sent to the nearest server that has the least workload and a soft transaction is sent to the highest energy server that has the least workload for transaction processing.

However, if the transaction is an UCV transaction, we need the most recent current value of the requested data item for transaction processing. The most recent current value of a data item usually resides at its primary copy server. However, if the coordinating server is not connected to the primary copy server as in this case, the transaction cannot be executed successfully. Hence, the coordinating server will try to connect to the primary copy server unless the deadline of this transaction has expired, in which case the transaction is aborted. The deadline here means the first deadline if the transaction is firm and the second deadline if the transaction is soft.

E. How to Synchronize Replicas?

Every time during the relocation period, the primary copy server of a data item tries to synchronize its data item with other connected servers that hold its replica. The primary copy server requests for the last updated timestamps from all replicas. Based on the update timestamp of the primary copy and the update timestamps of the replicas, the primary copy server determines if there is any other server that has a more recent value of its data item. If such a server exists, the new value of the data item is synchronized with all other servers. However, a server that is disconnected from the network during the relocation period cannot synchronize its data item. Even if the disconnected server has the most recent value of the data item, the primary copy server cannot determine it since the former is disconnected. Hence, it will only try to synchronize the data item during the next relocation period.

V. PERFORMANCE EVALUATION - PROTOTYPE MODEL

After considering the various open source database servers and clients based on our application requirements, we have chosen MySQL [16] server on Linux as the framework for our server database and DALP (Database Access Libraries for PDA) [2] on Windows CE as the framework for our client database system. We have modified the OLSR routing protocol in Linux to route packets and broadcast additional information like the energy and position of each mobile host. We have used a

Global Positioning System (GPS) to track the locations of mobile hosts that are used for both routing packets and for efficient transaction processing. We have modified MySQL and DALP to include our algorithm, DREAM. We have used this prototype to compare DREAM with the replication technique proposed in ([5], [6]), which we call the Hara's model and the "No Replication" baseline model. Sections 5.1 and 5.2 describe the client and the server transaction workflow in our prototype, respectively, and Section 5.3 describes the test database application.

Table I: Static Parameters

Parameters	Value
Memory_Size_Server	512 MB [3]
Memory_Size_Client	127 MB [10]
Processor_Speed_Server	2.8 GHz, 5503 MIPS [3]
Processor_Speed_Client	200MHz, 350 MIPS [10]
Battery_Life_Server	3hrs [3]
Battery_Life_Client	2.5hrs [10]
Bandwidth_Server	11 Mbps [14]
Bandwidth_Client	11 Mbps [1]

Table II: Dynamic Parameters

Parameters	Default value	Range
IAT (Inter Arrival Time between transactions)	0.5	Expon (0.1 – 1) [13]
Firm Transaction Probability	0.5	0, .25, .5, .75, 1
MRV Transaction Probability	0.34	0, .25, .5, .75, 1
MRVP Transactions Probability	0.33	0, .25, .5, .75, 1
OD Transactions Probability	0.33	0, .25, .5, .75, 1
Temporal Data Probability	0.4	0, .25, .5, .75, 1
Number of Partitions	2	1, 2, 3
MH disconnection probability	0.5	PROB(0.1) TO PROB(.9) [13]
Broadcast Frequency	2 sec	2- 100 sec
Relocation Time	256 sec	1 – 8192 sec [5]
Access Frequency Characteristics (p_i)	0.5	Zip parameter 0 - .99 [9]
Periodic Update Frequency	100 sec	0 – 300 sec [6]

The deadline of a transaction is computed based on its run time estimate and slack factor. A soft transaction's second deadline is twice as that of its first deadline. The performance is measured in terms of the percentage of transactions successfully executed, energy consumption of servers and clients, and the average difference in energy consumption between two servers. This third metric measures the balance of energy consumption distribution among servers. The static and the dynamic parameters used in our prototype model are shown in Tables 1 and 2, respectively. The impacts of the following five dynamic parameters are reported in this paper: firm/soft transaction ratio, transaction inter-arrival time, probability of mobile host disconnection, number of network partitions and broadcast interval.

A. Client Side

Each client is provided with an easy to use interface to generate real time transactions and associate appropriate deadlines to them. DALP provides a set of application programming interfaces that allows handheld devices to connect to MySQL databases. The client sends its firm transactions to the nearest server that has the least workload for processing to minimize the chance of missing their deadlines, while it sends its soft transactions to the highest energy server that has the least workload for professing so that a balance of energy consumption of mobile nodes can be achieved. When the transaction execution is completed, the client receives the transaction results and displays them to the end user.

B. Server Side

The database server after receiving a transaction fdetermines if it is a global transaction using its global conceptual schema. Distributed transaction processor functionality has been added to the MySQL database server, which divides transactions into sub-transactions and forwards these sub-transactions to appropriate participating server(s) that holds the required data. The local transaction processor then forwards these transactions to the real time scheduler that we have built into the MySQL server. The real time scheduler schedules transactions with shorter deadlines for execution before those with longer deadlines. The MySQL server has also modified to include a commit protocol that decides to abort or commit the transaction after communicating with the participating servers. After executing the commit protocol, the MySQL server sends the results back to the client.

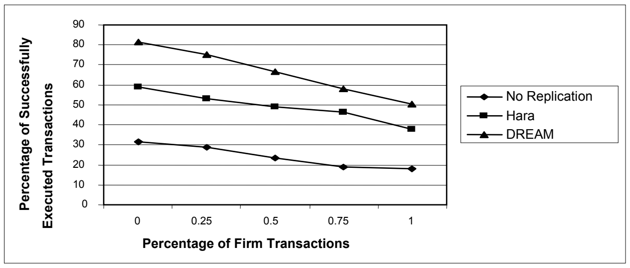
C. Test Database Application

We have obtained the data and transaction requirements for a military database application from the Reserve Officers Training Corps (ROTC) at the University of Oklahoma. We have created relational database tables for this application, populated each table with one million rows of data, and generated transactions to retrieve and update the data in the tables.

VI. PERFORMANCE EVALUATION RESULTS

A. Impact of Firm/Soft Transaction Ratio

The impact of the firm/soft transaction ratio on the percentage of successfully executed transactions and the server energy consumption is shown in Figure 6. More transactions miss their deadlines as the ratio of firm to soft transactions increases. Transactions with longer deadlines have more time to be processed and, hence, the probability of successfully executing such transactions is high. DREAM gives more priority for data items that are accessed frequently by firm transactions than those that are accessed frequently by soft transactions. Hence, DREAM has more successfully executed transactions. Consequently, the power consumption of servers in DREAM is the highest. However, the difference in server energy consumption between DREAM and the other two models is considerably lower compared to the difference in the number of successfully executed transactions.



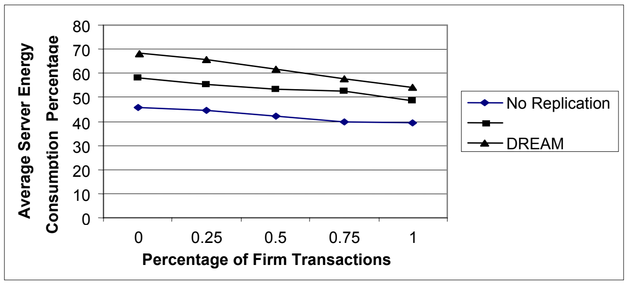
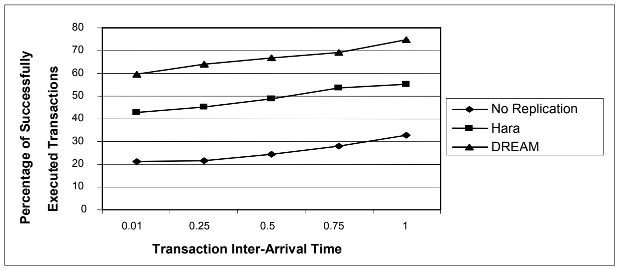


Figure 6: Impact of Firm/Soft Transaction Ratio

B. Impact of Transaction Inter-Arrival Time

In this experiment, the effect of system workload is studied by varying the transaction inter-arrival time. As transaction inter-arrival time increases, the speed of generating transactions decreases and, hence, the system workload decreases. Thus, transactions have less waiting time for resources and, subsequently, have a higher probability for successful execution. Since DREAM successfully executes more transactions, the power consumption of servers in this model is more than that in the other two models. However, Figure 7 also shows that the additional amount of energy that DREAM requires from servers is much less than the gain it makes in terms of the number of transactions successfully executed.



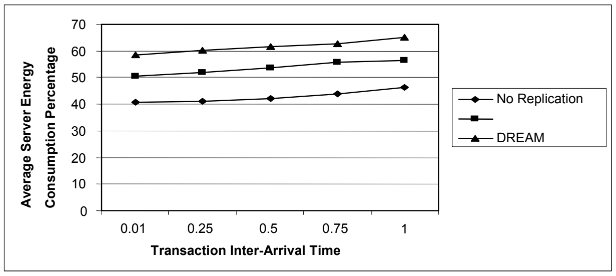
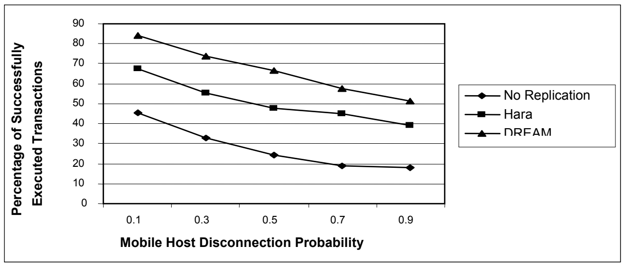


Figure 7: Impacts of Transaction Inter-Arrival Time

C. Impact of Disconnection Probability



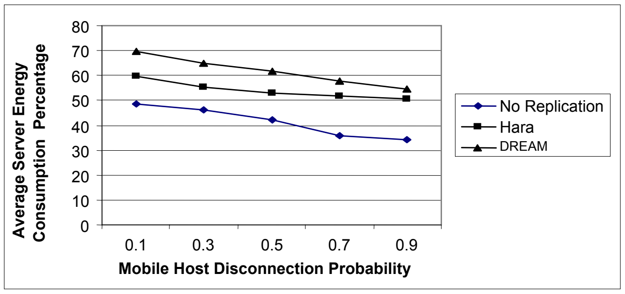
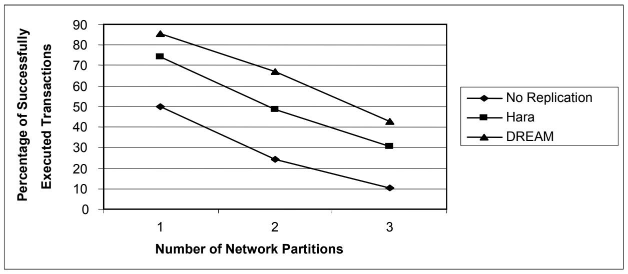


Figure 8: Impact of Disconnection Probability

In this experiment, the impact of mobile hosts' disconnection is studied by varying the mobile hosts' disconnection probability. When the disconnection probability is 0.5, the servers are kept out of each other's transmission ranges for 50% of the entire experimental run. When the probability for disconnection increases, the probability for nodes to be in different network partitions also increases. Thus, some servers might not be able to provide data services to clients that are in a different partition. Hence, the number of successfully executed transactions decreases with the increase in the probability of mobile hosts' disconnection as seen in Figure 8. As expected, the power consumption of clients is the maximum in DREAM as it successfully executes the most transactions among all the three models. But DREAM yields the most balance in energy consumption distribution among servers.

D. Impact of Number of Network Partitions

The impact of the number of network partitions is shown in Figure 9. As the number of network partitions increases, the number of successfully executed transactions decreases as servers in one network partition cannot provide data services to clients/servers in other network partitions. Server energy consumption distribution becomes less balanced with the increase in the number of network partitions as the isolated servers which host hot data items consume higher power than the other isolated servers



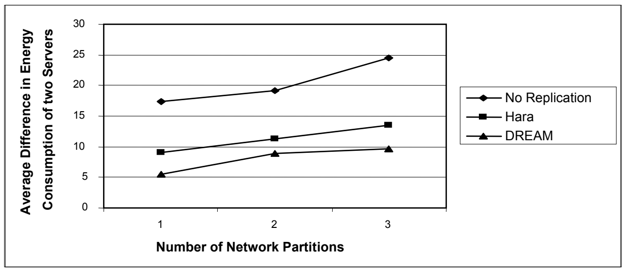


Figure 9: Impact of Number of Network Partitions

VII. CONCLUSIONS AND FUTURE WORK

A data replication technique called DREAM for real-time mobile ad-hoc network database systems was proposed in this paper. By replicating hot data items at appropriate servers based on the data model, real-time transaction model, read/write transaction model, current network topology, stability of wireless links, data access frequencies, and servers' remaining power, DREAM was demonstrated to perform the best in terms of percentage of successful transactions, energy consumption and distribution among nodes. As part of our future research, we plan to extend DREAM for group-based MANET architectures. We will also combine data caching and data replication for further improvement.

REFERENCES

[1] Cisco, http://www.cisco.com, Nov. 2004.
[2] DALP, http://www.kalpadrum.com/dalp, Nov 2004
[3] Dell, "Dell Users Guide", 2004.
[4] IETF MANET Home Page - http://ietf.org/html.charters/manet-charter.html, Dec 2003.
[5] T. Hara, "Replica Allocation Methods in Ad Hoc Networks with Data Update", *ACM-Kluwer Journal on Mobile Networks and Applications*, Vol. 8, No. 4, Aug. 2003, pp. 343-354.
[6] T. Hara, Y.H.Loh, S.Nishio, "Data Replication Methods Based on the Stability of Radio Links in Ad Hoc Networks", *Journal of the Information Processing Society of Japan*, Vol.44, No.9, Sept. 2003, pp.2308-2319.
[7] M. Hauspie, D. Simplot, J. Carle, "Replication decision algorithm based on link evaluation services in MANET", Technical Report, LIFL Univ, May 2002.
[8] X. Hong, M.Gerla, G.Pei, and C.Chiang, "A group mobility model for ad-hoc wireless networks," *ACM/IEEE MSWiM*, Seattle,Washington, 1999, pp.53-60.
[9] J. Gray, P. Helland, P. O'Neil, and D. Shasha, "The Dangers of Replication and a Solution", *ACM SIGMOD Int. Conference on Management of Data*, June 1996, pp. 173-182.
[10] Hewlett-Packard Laboratories, "Energy Management on Handheld Devices", Nov. 2004.
[11] G. Karumanchi, S. Muralidharan, R. Prakash, "Information Dissemination in Partitionable Mobile Ad Hoc Networks", *Symposium on Reliable Distributed Systems*, 1999, pp. 4-13.
[12] C. N. Lau, "Handling mobile host disconnection, data caching, and data replication in managing real-time transactions for mobile ad-hoc network databases", Master Thesis, University of Oklahoma, Aug. 2002.
[13] Y.Li, "Data Caching in Mobile Ad-Hoc Databases", Master Thesis, The University of Oklahoma, May 2004.
[14]Lucent, lucent.com/press/0400/000404.nsa.html, 04
[15] J.Luo, J.Pierre, H.P.Eugster,"PAN:Providing reliable storage in ad-hoc networks with probabilistic quorum systems", *ACM/SIGMOBILE Symposium on Mobile Ad Hoc Networking & Computing*, 2003, pp. 1-12.
[16] MySQL Open Source Database Server, http://www.mysql.com, Nov. 2004.
[17] P. Padmanabhan, "DREAM: Data Replication in Ad-Hoc Mobile Network Databases", Master's Thesis, University of Oklahoma, Dec. 2004.
[17b] P. Padmannabhan, L. Gruenwald, "DREAM: A Data Replication for Real-Time Ad-Hoc Mobile Network Databases," *IEEE Int. Conference on Data Engineering*, April 2006, pp. 134-137.
[18] D. Ratner, P. Reiher, G.J. Popek, "Roam: A Scalable Replication System for Mobility", *Mobile Networks and Applications,* Vol. 9 No. 5, Oct. 2004.
[19] B.Xu, O.Wolfson, S.Chamberlain, Y.Yesha, "Adaptive Lazy Replication in Unreliable Broadcast Networks", *Conference on Extending Database Technology,*2000.