DREAM: A Data Replication Technique for Real-Time Mobile Ad-hoc Network Databases*

Prasanna Pabmanabhan Le Gruenwald
School of Computer Science, The University of Oklahoma
prasannap@ou.edu, ggruenwald@ou.edu

Abstract

In a Mobile Ad-hoc Network (MANET), due to the mobility and energy limitations of nodes, disconnection and network partitioning occur frequently. In addition, transactions in many MANET database applications have time constraints. In this paper, a Data REplication technique for real-time Ad-hoc Mobile databases (DREAM) that addresses all these issues is proposed. DREAM is prototyped on laptops and PDAs and compared with two existing replication techniques using a military database application.

1. Introduction

MANET is a collection of wireless autonomous nodes that may move unpredictably, forming a temporary network without any fixed backbone infrastructure. All the nodes in MANET are mobile, power restricted, and thus, disconnection may occur frequently, causing a lot of network partitioning. Moreover many applications in this environment are time-critical and, hence, their transactions should be executed not only correctly, but also within their deadlines.

To improve data availability and system dependability, data in a client-server MANET system should be replicated at various servers. However, no single existing replication technique considers all of the above issues. To fill this gap, this paper proposes a data replication technique, called DREAM. The rest of the paper is organized as follows. Section 2 describes the MANET database architecture that is used in DREAM. Sections 3 and 4 present DREAM and its prototype results, respectively. Finally, Section 5 concludes the paper with future work.

2. Decentralized MANET Architecture

Depending on communication strength, computing power, and storage size, mobile hosts are classified into clients, which store only the query processing module of the Database Management System (DBMS) that allows them to submit transactions and receive results, and servers, which store the complete DBMS and provide data services to clients. In a decentralized architecture, clients can communicate (single-hop or multi-hop) and submit their transactions to any of the available servers. This architecture does not place reliance on any centralized server, and thus, improves system resilience.

3. Proposed Replication Technique: DREAM

DREAM is composed of three main parts. The first part determines the data items 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.

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. Since firm transactions must be aborted if they missed their deadlines, while soft transactions can still be executed even after their deadlines have expired, DREAM offers 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. 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 (replica) servers.

3.1. Data Types

Data items are classified into read-only and read-write data items. Read-write data items are further classified into temporal and persistent data items. The values of the former are valid only for a certain period of time, called *age*, while those of the latter are valid throughout their

This material is based upon work supported by (while serving at) the National Science Foundation (NSF) and the NSF Grant No. IIS-0312746. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

existence. All read-write data items are further classified into periodic and aperiodic update data items.

3.2. Transaction Types

Transactions, both firm and soft, are classified as read and write transactions. *Read transactions* are further classified into: 1) Most Recent Value (MRV): transactions that need the most recent values of data items across all database servers in the network; 2) Outdated transactions (OD): transactions that can be executed successfully even with stale data; and 3) Most Recent Value in Partition (MRVP): transactions that require the most recent values of data items across all servers in a partition. The result of a MRVP transaction may or may not be the most recent value of the requested data across all database servers.

Write transactions are classified further into: 1) Insert/Delete: transactions that insert or delete records into the database; 2) Use Current Value (UCV): transactions that need the current values of data items for processing; and 3) Overwrite Current Value (OCV): transactions that overwrite the current values of data items.

3.3. What and Where to Replicate?

The first step of the algorithm is to calculate the weighted access frequencies of data items based on their data and transaction types. The next step is to replicate data items with higher weighted access frequencies in servers that have the maximum remaining power. After storing the 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 is taken every time during a certain period of time called the *relocation period* [2].

3.3.1. Computing Weighted Access Frequencies

Access frequency of a data item *d* at a particular server *s*, $Access\_Frequency_d^s$, is the number of times that *d* is accessed at *s*. From the access logs, the access frequency of each data item at each server is computed [2]. In addition, in DREAM, the numbers of times a data item is accessed by Firm, Soft, MRV and Non-UCV transactions are computed. These access frequencies are further calculated using a weight factor based on the data and transaction types as follows.

a) Firm and Soft Transactions: if a data item is accessed more often by firm transactions than another data item, then the former is given a higher priority for replication. This is to reduce the number of firm transaction aborts due to deadline expiry.
b) Temporal Data Item: If the age of a temporal data item is higher than the relocation period, the probability for successfully accessing that data item until the next relocation period is high. The ratio between the remaining valid time interval and the relocation period is called the *Age Relocation Ratio (ARR)*. Data items with higher ARR are replicated before the ones with lower ARR.
c) Read-Write Data Item: Data items that are accessed frequently by UCV transactions are given a lower priority to be replicated than the ones that are accessed frequently by Non-UCV transactions.

3.3.2. Replica Allocation and Redundancy Elimination

In DREAM, the link stability connecting two servers until the next relocation period is given by the formula:

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

A higher RR between two servers means that the link connecting them is more stable. After determining the access frequencies and the RR 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 are similar, the same data items will be replicated [2]. Replica duplication among neighboring servers can be eliminated by replacing a duplicate data item with the next highest accessed data item as presented in [2]. However, if those neighboring servers get disconnected, data accessibility would even decrease because of such a replacement. Hence, 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 is based on the following 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 in s_i. There is, however, no use in replacing d_x by d_y if the difference between their access frequencies in s_j is very high or if the link connecting s_i and s_j is unreliable. 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 and the number of times that d_x can be accessed from s_i is greater than the number of times that d_x can be accessed from s_j.
b) Both of the servers are secondary copy servers of d_x: in this case, 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 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.

3.4. How to Access Replicas?

When there is a request for a data item *d* to its primary copy server *p*, it can be accessed directly. If the initiated transaction is a write transaction, *p* broadcasts its update timestamp to indicate to the secondary copy servers that their replicas are out of synchronization. In contrast, if the coordinating server is not the primary copy server, the data item is accessed based on the following criteria:

Read Transactions: if the requested data item is a read-only data item or if the initiated transaction is an OD transaction, it can be accessed from any server. 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. The objective is to reduce the number of transaction aborts and, at the same time, balance the energy consumption distribution among servers.

If the transaction is a MRV transaction, the requested data item should be accessed from the server that has its most recent value. 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 $Local\_Update\_Timestamp_d^s$ 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. Hence, a MRV/MRVP transaction accessing such a data item can read it from any server that has updated it during its last known update time interval.

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. 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 directly accessed.

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 the transaction is an UCV transaction, and if the coordinating server is not connected to the primary copy server, 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.

3.5. How to Synchronize Replicas?

Every time during the relocation period, the primary copy server of a data item tries to synchronize with other secondary copy servers. The primary copy server requests 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 [5]. 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.

4. Prototype Model

After considering the various open source database servers and clients based on our application requirements, we have chosen MySQL [4] server on Linux as the framework for our server database and DALP (Database Access Libraries for PDA) [1] 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.

Clients generate real time transactions and associate deadlines to them. A server after receiving a transaction determines if it is a global transaction using its global schema. Distributed transaction processor functionality has been added to the MySQL server, which divides transactions into sub-transactions and forwards them to appropriate participating server(s). The local transaction processor then forwards these transactions to the real time scheduler that we have built into the MySQL server. It schedules transactions with shorter deadlines for execution before those with longer deadlines. A commit protocol that decides to abort or commit the global transaction was also added to the MySQL server.

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.

We have used this prototype to compare DREAM with the “No Replication” model and Hara’s replication technique proposed in [2, 3]. The former is selected as it

illustrates the impact of having no replication in MANET databases. The latter is chosen as DREAM is built on some of its ideas. In Hara's technique, data items with higher access frequencies are replicated before cold data items. It also detects network partitioning and replicate hot data items before such a partitioning occurs to improve data accessibility. However in this technique, 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.

The impact of the firm/soft transaction ratio on the percentage of successfully executed transactions and the average energy consumption distribution among servers is shown in Figure 1. 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 data items that are accessed frequently by soft transactions. Hence, DREAM has more successfully executed transactions than the other models. 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 between DREAM and the other two models. This means that with the same number of transactions successfully executed, DREAM would require less server energy than the other two models. It can also be seen in the 2nd part of Figure 1 that DREAM is the most balanced model in terms of energy consumption distribution among servers. This is because DREAM replicates hot data items at servers that have the most remaining available energy.

The impact of mobile hosts' disconnection on the performances of the data replication algorithms when the mobile hosts' disconnection probability is varied from 0.1 to 0.9 is shown in Figure 2. 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 mobile hosts to be in different network partitions also increases. Since some servers might not be able to provide data services to clients that are in a different network partition, the number of successfully executed transactions decreases with the increase in the probability of mobile hosts' disconnection. As expected, the power consumption of clients is the highest 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.

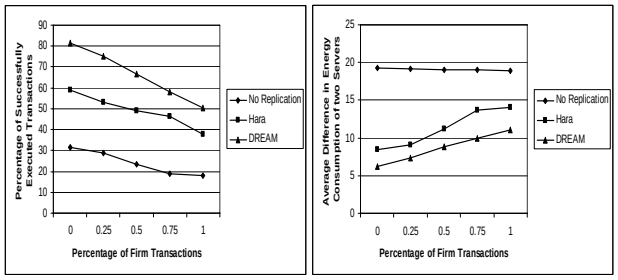

Figure 1: Impact of Firm/Soft Transaction Ratio

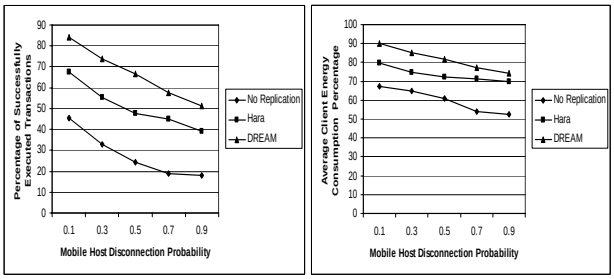

Figure 2: Impact of Disconnection Probability

5. 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, 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 successfully executed transactions and balance of energy consumption distribution among servers. As part of our future research, we plan to combine data caching with DREAM and extend it for group-based MANET architectures.

References

[1] DALP, http://www.kalpadrum.com/dalp, Nov 2004

[2] 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.

[3] 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.

[4] MySQL Open Source Database Server, http://www.mysql.com, Nov 2004.

[5] P. Padmanabhan, "DREAM: Data Replication in Ad-Hoc Mobile Network Databases", Master's Thesis, University of Oklahoma, Dec. 2004.