Energy Efficient Data Replication for Mobile Ad hoc Networks

Abstract:

Mobile computing is becoming need of the time and its increasing use has compelled to explore the possibilities of efficient, reliable and cost effective resources of such computing. Mobile hosts run on limited energy source (i.e. battery power) and communicate with each other via wireless links of limited bandwidth. Hence, it is necessary to incur low communication cost during the transfer of information from one mobile host to another. In an ad hoc environment, ensuring reliability of obtained information is a major problem due to frequent partitioning of the network as a result of the mobility of servers. It becomes absolutely necessary to improve the processing time for transactions while respecting battery power considerations and ensuring the reliability of information. Data Replication in Ad hoc mobile databases is gaining importance due to the need for an effective replication scheme and various complications involved in replicating data for ad hoc systems. Efficient data replication among the energy sufficient hosts could result in providing better and reliable source to database queries originating from mobile hosts. Available techniques [1] [2] give solution for replication in mobile ad hoc networks with compromise on time and consistency respectively. An analysis with simulation of present available techniques for data replication in mobile ad hoc networks and possible modifications / Improvements are proposed.

1. Introduction:

A "mobile ad hoc network" (MANET) is an autonomous system of mobile hosts connected by wireless links, union of which form an arbitrary graph. The hosts are free to move randomly and organize themselves arbitrarily; thus the network's wireless topology may change rapidly and unpredictably resulting in appearance of network partitions in infrastructure-less, or ad hoc networks. Mobility imposes lot more constraints on the performance of network than in static or stationary network system.

In general replication aims towards the efficiency of overall system. i.e. by providing a replica at a secondary site we can cut on the communication cost as well as the processing time of the transaction.
Replication aims in mobile environments are

- *high availability* of data on mobile sites during Network Partitions,
- *consistency* despite of partitions
- *low communication costs*.

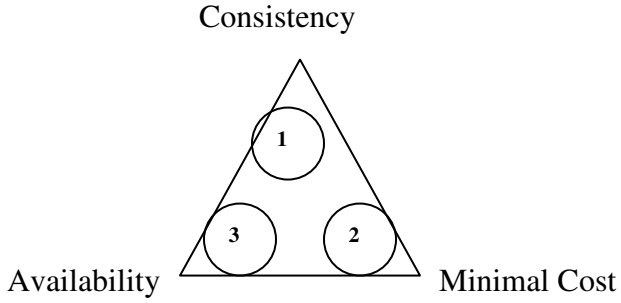

Fig 1

These objectives span a triangle illustrating in Fig. 1, the impossibility of achieving the three objectives simultaneously. Attaining one objective requires making cuts in at least one of the two remaining objectives. For example, a network could be made to provide consistent data. Reducing availability on the other hand can attain this. Simply, having fewer copies means less possible inconsistencies, hence fewer immediate modification propagations, which increase costs. If cost reduction is a primary concern and is achieved infrequent and short connections, consistency and availability are not attainable.

In mobile ad hoc networks, the nodes does not have unlimited power source unlike the fixed hosts, so the limited energy resources of the mobile hosts and clients has to be consider as a *vital factor* in their performance metrics. Keeping the scenario of compromising one of the characteristics of efficiency, efforts are made to provide a solution, which could enhance the availability with reasonable communication cost and vigilant usage of battery power. Recent developments in design of microprocessor and peripheral devices are strongly addressing the power consumption, size and speed issues.

Data replication is aimed to make the data locally available, hence reducing cost in terms of communication and energy. Generally, data replication is efficient for systems having less frequent updates than reads, since updating replicas is expensive. In our model we have assumed the read frequencies are much higher than update frequencies of database

Most present approaches offer a range of concepts from restricting the mobile work over lazy consistency to mobile work reserved. Approaches distinguishing updateable core copies for read- and write-access and read-only cached copies are unacceptable for meeting the requirement of synchronized replicas. Numerous applications for mobile users work isolated, but require more than cached copies and a finer granularity than files.

2. Overview of existing related techniques:

Traditional replication scheme are based on assumptions like hosts being fixed and access patterns being static. Since these assumptions no longer in a mobile ad hoc environment, so new replication schemes have to be introduced.

Eager Replication attains replication by keeping all replicas exactly synchronized at all nodes. This in turn is achieved by updating all replicas in one transaction. No concurrency anomalies occur since eager replication gives serialized execution. Eager replication is not a proper choice for a mobile environment since eager replication reduces update performance with instant updates, and in a mobile environment the nodes are not accessible all the time.

Lazy replication algorithms are more suited to the mobile environment. Here, the synchronization occurs after the updating transaction commits. That is, replication is asynchronous in nature. The problem with lazy replication is that it *may allow* a transaction to see a very old committed value.

Presently, several proposed replication schemes for mobile environments are in use. Some of these are:

- Replication via dynamic data allocation.
- Adaptive data replication
- Active replication.
- Optimistic replication.
- User Profile replication

2.1. Replication through Data Allocation [Huang, 1994]

Replication is needed when a mobile user frequently accesses data that is updated infrequently. Allocation of a copy of such data to the mobile user can reduce costs involved in such transactions. This leads to the formulation of two allocation schemes: *one-copy* and *two-copy* schemes. In the one-copy scheme, only the stationary computer has a copy of the data, whereas in the two-copy scheme, both the mobile-user and the stationary computer have a copy of the data.

Data allocation schemes can be *static* or *dynamic*. The dynamic data allocation method of [Huang, 1994], switches between one-copy and two-copy schemes, depending upon the read/write ratio. Each time the mobile user tries to read data, the latest *k* numbers of requests are examined. If in these 'k' requests, the number of reads is more than the number of writes and the mobile user does not possess a copy of the

data it is trying to read, then such a copy is allocated to the mobile user. On the other hand, if the number of writes exceeded the number of reads and the mobile user has a copy of the data it is trying to read, then such a copy is de-allocated. Such a dynamic allocation scheme is referred to as a sliding window scheme [Huang, 1994]. It uses either one-copy or two copy scheme.

2.2. Active Replication Scheme [Wu 1998]

In mobile environments, since mobile users have no fixed locations, access patterns change as the mobile users move. Hence, replication schemes have to dynamically change the replication structure so as to have the latest access patterns. This also allows for a dynamic recalculation of cost. Dynamic replication schemes are not actually dynamic in the sense that the replication structure reconfiguration occurs only after the changes in the access patterns have taken over a period of time and reflected on the access statistics. This should intuitively lead to a replication scheme that is predictive in nature. That is, the scheme should be able to derive the mobile user's mobility pattern and data requirement based on some information provided by the mobile user or collected from the mobile user. The active replication scheme proposes to achieve this kind of predictive replication.

This scheme makes use of user profiles to obtain the mobility patterns, access behavior and read/write patterns. It then uses this information to derive the users' mobility patterns and data requirements and actively reconfigures the replicas to adjust to the changes. Each user's profile contains a detailed access statistic.

Each user is also required to provide his/her daily schedule. This is done so that the replication decision that is to be made matches accurately the movement and data requirement of each user. This allows allocation of replicas before the actual access. The scheme makes use of the concept of *open objects* [Wu 1998] to represent the user's present and future data requirement. A user's access statistics are correlated to the read write requests made on data objects.

At any time and location, a user never accesses the entire subset of data objects that are available. In fact, a user will access a subset of the available data set. Open objects are defined to be the *set of objects accessed since entering the current cell* [Wu 1998]. Open objects can be used to calculate the access cost because they provide a portal to what data may be accessed in the future. Hence, open objects contribute to the predictive nature of the active replication scheme.

The basic reason for allocating a replica is to minimize the cost of accessing data from a site that is remote. The criterion followed is that if the cost of satisfying a read request is greater than or equal to the

cost of satisfying a write request to the same site, then it is justifiable to maintain a replica at the site under question. In a scenario other than the one mentioned above, it would not be feasible to maintain the replica. This is probably due to the cost of maintaining the replica itself would undermine the aim of minimizing the cost of accessing the data.

The effectiveness of the active replication scheme is based on the availability and location of the mobile users. If the users move around in a fixed geographical area, then the predictive nature of the active replication scheme kicks in. Since the users mobility pattern is now fixed, the prediction of their data access characteristics can be made accurately to a larger extent. Also, the replicas are going to be allocated close together and this would aid in reducing the access cost.

2.3. Virtual Primary Copy Replication (VPC) Method [Zaslavsky 1996]

The approach to the virtual primary copy method is to always maintain a copy of the primary copy on a fixed host. Such a copy is referred to as a virtual primary copy. Suppose the mobile host holds the actual primary copy but is disconnected from the network. Then, the virtual primary copy comes into the picture and impersonates the functions of the actual primary copy. Any host trying to access the primary copy actually transacts with the virtual primary copy, being under the impression that it is transacting with the actual primary copy.

A mobile host is required to register with a home or base node. It is the job of this home node to see that the actual primary copy and the virtual primary copy remain synchronized. Also, the home node is usually used to host the virtual primary copy. The reasoning behind this is that the mobile host will contact its home node more often than any other node.

The virtual primary copy method bears the overhead of synchronizing the primary copy and the virtual primary copy when compared to the primary copy method in which the primary copy is stored on the mobile host. The processing cost reduces in the virtual primary copy method, as when the mobile primary copy is unavailable; the virtual primary copy is accessed, which is present on the fixed host. In the primary copy method, the data would have to be accessed by requesting for it over the costly wireless medium. It should be noted that there would be an additional processing cost involved during synchronization between the virtual primary copy and the primary copy. Looking at the average response time of the two methods, it was found that the virtual primary copy method has a better response time when the transactions to be carried out in a given period are fixed [Zaslavsky, 1996].

The primary copy method could cause a problem in a large system. If frequent updates take place, then it could lead to a bottleneck situation, as all write transactions would lead to the node containing the primary copy. The virtual primary copy method would perform better in such a situation [Zaslavsky, 1996].

The virtual primary copy method is preferred in applications where throughput is not a main concern. On the other hand, the primary copy method is preferred in read-oriented applications because of the better response time it provides [Zaslavsky, 1996].

Since the virtual primary copy method has the overhead of resynchronization, it is important to make use of efficient resynchronization algorithms to reduce the cost involved in this process.

2.4. Three-copy Replication Scheme [Lee 1998]

The three-copy replication scheme is based on the optimistic replication strategy. It proposes a hierarchy structure and a client-agent-server architecture, which is more suited to fit into a mobile computing environment. This scheme based on the strategy that a mobile host cache useful data or data that it accesses frequently. Then, when it is disconnected from the network, it can still continue processing a request using the data that is present locally in its cache. Hence, this would result in the saving of a wireless link between a database server and the mobile host.

The agent-based architecture proposed in this scheme has a mobile host, a fixed foreign agent, and a fixed database server. Other possible architectures could have a mobile host, a fixed foreign agent and a mobile database server, or they could have all three mobile, that is, a mobile host, foreign agent and database server. Agent plays an important role in that, if an agent contains some information in its cache that a mobile host needs, the mobile host can access it from the agent rather than access from it the database server, thus reducing communication time between the database server and the mobile host.

Conflict handling is performed in two ways. The database server handles global conflicts (occurring for mobile hosts under different agents, but the same database server). The foreign agents handle local conflicts (occurring for mobile hosts under the same foreign agents and database server). When the mobile host moves from one cell to the other, it registers with the base station of the new cell. It then looks for the foreign agent whose jurisdiction is the new cell. When such a foreign agent is found, the foreign agent of the previous cell (that is, the cell from which the mobile host just moved) forwards cache data and the mobile host's profile data to the new foreign agent.

Update occurs at both the primary copy and the mobile copy, and the update occurs first on the primary copy. This is essentially a conflict and is resolved as follows. A view of the time stamp would indicate that the update occurred first on the primary copy. A mobile host would then rollback its transaction and synchronizes with the primary copy. Such a transaction is referred to as a *compensating transaction*. It would then re-execute its previous transactions.

Update occurs at both the primary copy and the mobile copy, and the update occurs first on the mobile copy. In such a scenario, the mobile hose needs to inform the database server as soon as possible, so that the database server can synchronize by making the required changes.

When these are effective?

The users in a mobile environment have changing access patterns that a static replication scheme does not keep track off. Traditional replication schemes were designed assuming that the hosts concerned would be fixed. Hence, these schemes were static and did not change with time.

Basically, the dynamic replication schemes change their replication scheme as the access pattern of the mobile user changes. They overcome the drawbacks of static replication and provide a cost-effective way to replicate in a mobile environment. Among the replication schemes for mobile data management, the ones that are *predictive* in nature have a higher performance and reduce the cost of operation.

3. Mobile Ad hoc Environment:

So far we have discussed the mobile environment in milieu of Data Replication. Since our primary objective is to analyze the data replication techniques in mobile ad hoc networks. In the following section we will give a brief description of two [Karumanchi, 1999] [Gruenwald, 2000] primary techniques for our analysis in terms of efficient data replication in mobile ad hoc networks. A Simulation comparison is performed to compare the performance of GBRMAD and proposed Improved version of GBRMAD namely, MGBRMAD.

4. Quorum Based Replication Scheme [Karumanchi, 1999]

[Karumanchi, 1999] proposes a technique (QBS) for information dissemination in mobile ad hoc networks. This technique gives a quorum based solution for MANET and addresses the common problems arising in this atmosphere due to frequent partitions and changing network topology. For example:

- When and where to Update?
- Who to send a Query?

The model used by Karumanchi consists of *N* fire fighters and *n* officers who are managing the operation. With n << N. The firefighters have wireless communication devices to communicate with officers, who have somewhat larger units to serve as servers. The set of fire fighters constitute an ad-hoc network of *N* nodes. There is no a priori association between fire fighters and officers. Inherited from design of the network, it could face frequent partitions. A querying node will send his/her query to some officer without any prior knowledge of the mobility pattern of the node whose information it is seeking.

The proposed solution uses the variation in quorum-based scheme; additionally it uses the assumption of real-time applications where inaccurate information is preferable than to no information at all.

Given a set of *S* Servers, quorums are *m* subsets of *S*, namely S_0, S_1, S_2, …. S_{m-1}, such that

$$\bigcup_{i=0}^{m-1} Si = S$$

$$Si \bigcap Sj \neq \phi, 0 \leq i, j \leq m-1$$

All the sets Si were constructed a priori with known membership. Size of quorum was chosen to be $\sqrt{n}$. Servers form quorums such that together they form whole set of Servers and there is always at lease one server common in any two quorums (i.e. they are not disjoint).

It uses the *absolute connectivity based* strategy to address the question of when to update the position of the mobile units. Since the network partition is unpredictable, so a node will send an update when a certain pre-specified number of link incidents have been established or broken since last update. Other policies *(time-based, time and location based & percentage connectivity based strategies)* were also proposed but *absolute connectivity based* outperformed the rest in simulation experiments. Also each node X maintains a DQL_X , which gives X's perception of un-reachable hosts. Any sever who is unreachable will be there in DQL of X for a latency of δ_{DQL}, after which it will be removed from that list.

Querying node uses Random Selection (to balance the load) of a quorum for sending his/her request. There are three strategies proposed in the technique for performing the query/update transactions. Namely;

3.1.1 Select then Eliminate (STE):

- Node X will randomly select a quorum S_i
- X will send the request to all those of S_i, which are reachable ($S_i` = S_i - DQL_x$).
- Within timeout period $T_{timeout}$, if at lease a single response is received from a server, then update is considered successful, similarly for read/query if more than one non-Null reply is received then the one with higher time stamp is selected.
- If no response is received from a server for $T_{timeout}$ then that server is included in DQL_x for *disqualification duration* of $\delta_{DQL.}$
- This process is repeated until there is success.

This strategy tries to maximize the availability of information. Since all the nodes are available for update or query, thus increase the availability.

3.1.2 Eliminate then Select (ETS):

- Node X will first de-select all the quorums who have at least one node in DQL_x.
- X will randomly select one of the remaining quorums and send the request to servers in that quorum.

- If he receives at least one acknowledgment in case of update, and a non-Null reply in case of query, the transaction is success.
- If some of the servers do not response within $T_{timeout}$, while other send Null reply. Then the one, which did not send the reply, will be included in DQL_x.
- Quorums containing at least one member in DQLx are de-selected.
- One of the remaining quorums is selected for query / update request.
- This process is repeated until success.
- If no quorum left then transaction is aborted.

This technique aims to identify a quorum with no server in the DQL. This in result will reduce the number of available quorums available for updates at maximizing the number of servers receiving update/query.

3.1.3. Hybrid Strategy:

- Hybrid strategy makes use of both ETS and STE to take advantage of both strategies.
- It uses STE for Read queries
- Uses ETS for Updates.

With simulation result they found that Hybrid strategy has a performance edge over the individual techniques. Since it takes advantage of both the strategies. Fig. 2 explains the flow of transaction in this technique.

Transaction Flow Diagram for Quorum Based Solution

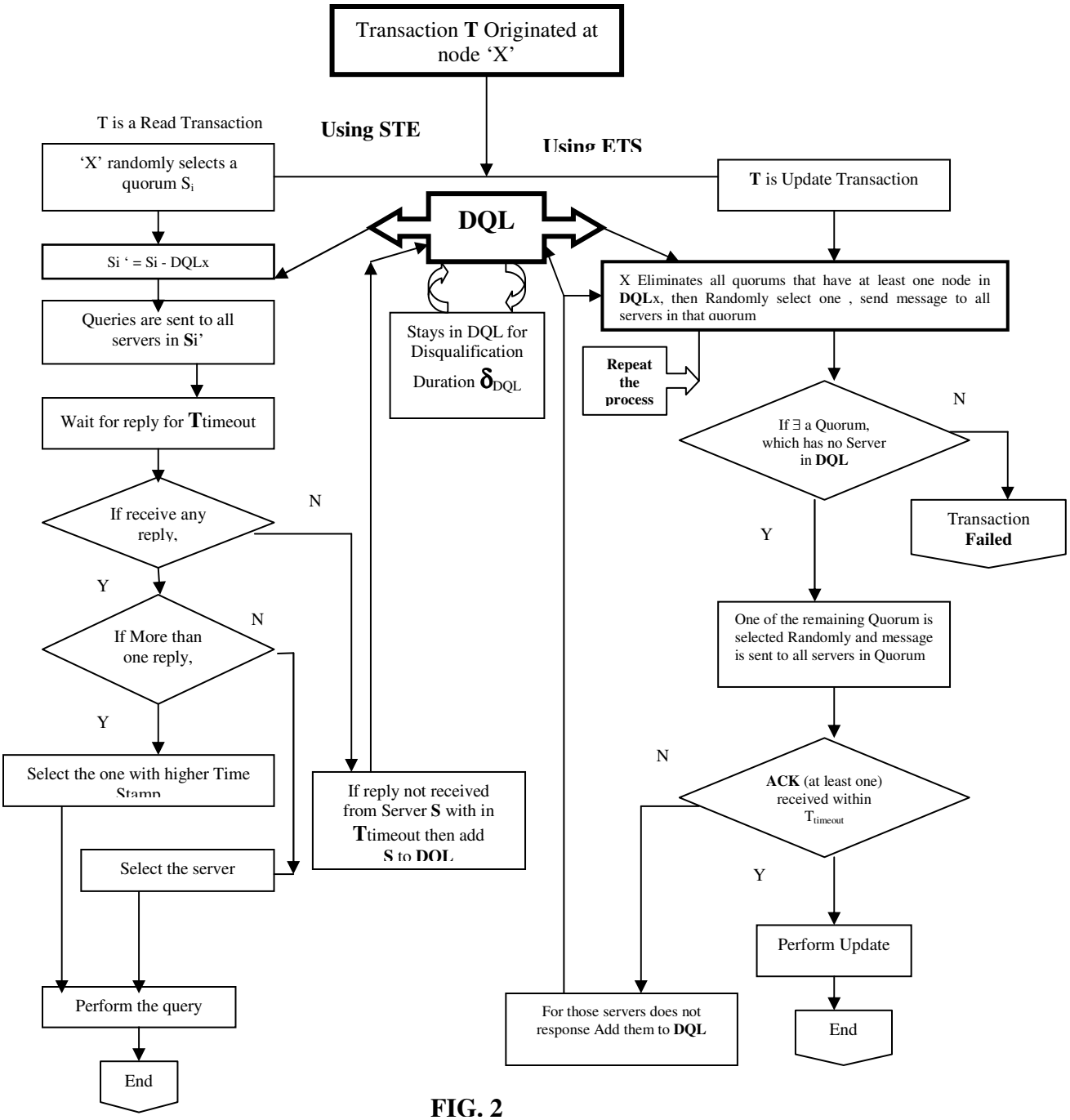

FIG. 2

3. 2. Group Based Replication Scheme:

[1] Proposes a technique (GBRMAD) for data replication in mobile ad hoc databases. This technique aims at achieving the accuracy of queries and decreasing the use of power of mobile nodes. The model [Gruenwald, 2000] used by GMRMAD is *quoted* below for reader's convenience.

3. 2. 1 System Model:

"*In ad-hoc networks, Mobile Hosts (MHs) communicate with each other without the help of a static wired infrastructure. These types of networks are usually used in battlefields. So, we have defined our architecture considering the battlefield environment as illustrated in Figure 1. Depending on communication capacity, computer power, disk storage, size of memory and energy limitation, MHs in the military network architecture can be classified into two groups: 1) computer with reduced memory, storage power and computing capabilities (e.g. soldiers equipped with portable computing and transceiver devices), which we will call Small Mobile Hosts (SMHs), and 2) classical workstations equipped with more storage, power, communication and computing facilities than the SMHs, which we will call Large Mobile Hosts (LMHs). These LMHs can be classified into two subgroups – humvees and tanks. Humvees have high capacity communication links and relatively stable. Tanks have less storage, computing and energy capacities and move often than humvees. Both humvees and tanks are more static than SMHs. Soldiers (i.e. SMHs) can communicate with tanks and humvees via wireless LAN technology. One soldier can talk to several humvees or tanks at the same time.*

Every MH has a radius of influence. In Figure 3, a circle shape with borders in dotted line represents the radius of influence of an MH. An MH can directly communicate with other MHs, which are within its radius of influence. The communication link between two MHs is shown with dark dotted lines in Figure 1. In our proposed environment, if two MHs are outside each other's radius of influence they will be able to indirectly communicate with each other in multiple hops using other intermediated MHs between them. For example, in Figure 1, SMH 11 will not be able to communicate directly with LMH 3 because their radii of influence are not overlapping, but it can indirectly communicate in multiple hops using SMH 10 and SMH 9 between them.

MHs in battlefields are not connected to unlimited power supplies and thus have energy limitation. To reduce the energy consumption, the MHs can operate in three modes- Active mode, Doze mode and Sleep mode.

1) *Active Mode: The MH performs its usual activities. Its CPU is working and its communication device can transmit and receive signals.*

2) Doze Mode: The CPU of the MH will be working on a lower rate. It can examine messages from other MHs. The communication device can receive signals. So the MH can be awaken by a message from other MHs.

3) Sleep Mode: Both the CPU and communication device of the MH are suspended.

Due to energy and storage limitations, we will assume that only LMHs will store the whole Database Management System (DBMS) and SMHs will store only some modules of the DBMS (e.g. Query Processor) that allow them to query their own data, submit transactions to LMHs and receive the results."

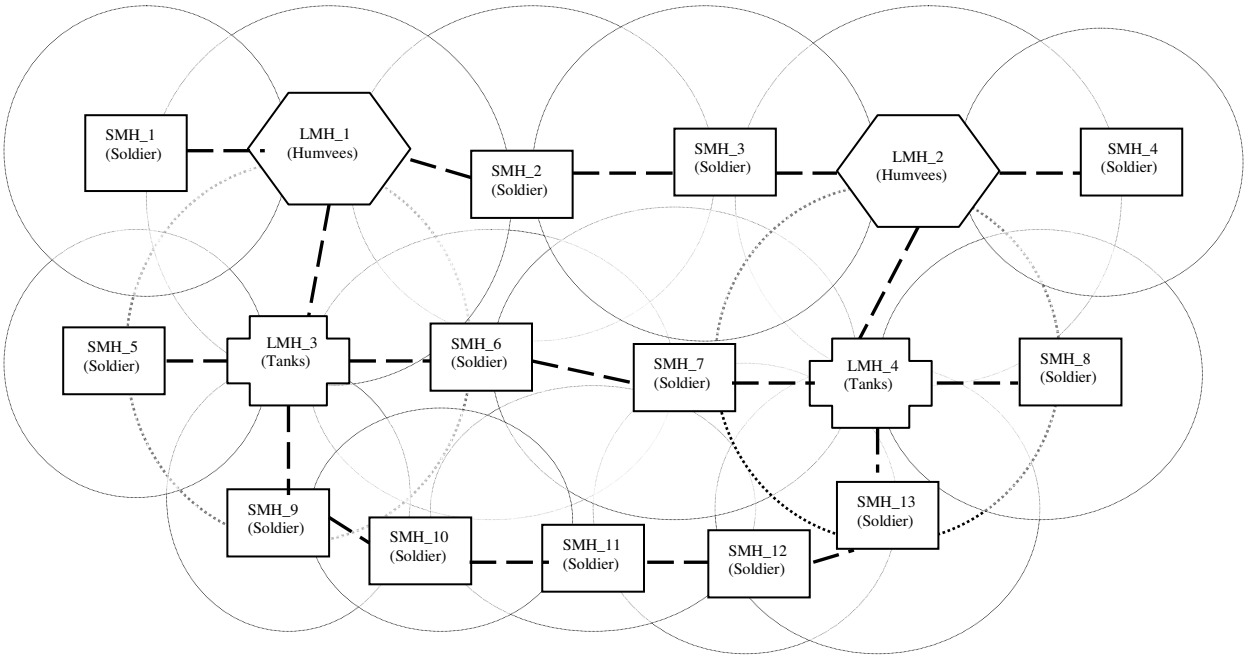

FIG 3. Architecture

3. 2. 2. Technique Details:

The proposed technique is named Group Based Replication for Mobile Ad Hoc Databases (**GBRMAD**) as described above [Gruenwald, 2000], aims at achieving the accuracy of queries and decreasing the energy consumption of the mobile nodes.

Following are the assumptions made in designing the model model.

- Transaction could be Firm / Soft
- Transactions could be MRV / Normal
- No. of Read accesses are higher than no. of updates.
- All transactions are compensate-able
- LMHs have fairly large storage capacity
- No. of read transactions for all LMHs are same.
- Initially, static data allocation is assumed and also the SMHs (Small Mobile Hosts) are pre-assigned with their corresponding LMH.

Let $S_1, S_2, \ldots\ldots, S_n$ be n LMHs (Large Mobile Hosts) with $C_1, C_2, \ldots\ldots, C_n$ and $E_1, E_2, \ldots\ldots E_n$ be the initial storage capacity and energy level of n servers, respectively. Let SG_i denote the i^{th} server group. The servers are divided into m groups such that:

1) $\bigcup_{i=1}^{m} SG_i = \text{Set of all servers}$

2) $SG_i \cap SG_j = \varnothing$, for $i \neq j$

- Any data item X will be primary for specific groups say SG_i. X_{prim} will be the primary copy residing at servers of SG_i. Dynamically allocated copy X_{sec} will be allocated to servers from any group SGj such that i≠j, where it is feasible to allocate a replica, details of which are discussed later.

- A database with k data item is divided into $\boldsymbol{m}$ Groups. Read access ratio RRi of each group i is then calculated and sorted in descending order.

$$\mathbf{RR}_i = (\mathbf{A}_i \ / \ {}^{r=m}\Sigma_{r=1} \mathbf{A}_r)$$

- The Groups with higher read access ratios will get more servers assigned. The process of allocating servers to groups is explained:

    - Each energy level of servers is considered as a class and servers with same energy level belong to same class. If total energy levels are J then J classes.
    - Let there are L energy levels resulting in L classes, then E_L be the energy level of each server in L^{th} class.
    - No. of Servers with a particular energy level, required for each group is calculated from the product of its read access ratio and the number of servers of any particular energy level. Mathematically,

$$N_{SGi}^{E_L} = \mathrm{RR}_i \times N(E_L)$$

    - The algorithm for assigning servers to groups assigns, first tentative number of servers of a given energy level to each server group. Following, remaining servers to be assigned to groups requiring additional servers. Idea behind is to distribute equal proportion of servers of different energy levels among all groups.

- Each data item is stored as a tuple < value, timestamp> and the one with higher time stamp is selected for read.

3. 2. 3. Read Transactions:

- A transaction Tm originated at server Sj READS the data item X. If Sj belongs to the primary group of X and Tm is a transaction requiring the MRV of X, a lock request is send to all the servers in the primary group of X. The transaction then waits for a pre-specified time period $T_{timeout}$ for the primary servers to respond.

    - If more than $\lceil N_{SGi} / 2 \rceil$ locks are obtained, Sj arranges all the primary servers in the increasing order of their distances from Sj. The data item X is then read from $\lceil N_{SGi} / 2 \rceil$ primary servers that are nearest to Sj. If exactly $\lceil N_{SGi} / 2 \rceil$ locks are obtained, data is immediately read from these servers. Reading data from $\lceil N_{SGi} / 2 \rceil$ ensures that the most recently updated value of the data item X is read. For a MRV transaction if less than $\lceil N_{SGi} / 2 \rceil$ locks are obtained the transaction is aborted.

- For a normal transaction, a lock is requested from a single primary server that is closest to Si. The data item X is then read from this primary server.
- A firm/soft transaction can either be a MRV or a normal transaction. If a firm transaction Tm misses its deadline, Tm is aborted. For soft transaction Tm, Tm continues to execute even if the deadline is missed.
- A transaction Tp originated at a secondary server Sk with respect to the data item X checks for the presence of the secondary copy of X at Sk. If no copy of exists at Sk and it is beneficial to allocate a replica of X at Sk, a replica is allocated at the server Sk. The replica is obtained from the primary copies of X at the primary servers on which locks are obtained. The ***most recent*** value of all the primary copies is taken as the value of the replica. Tp then reads the data item X locally.
- If a copy exists at Sk, Tp reads the data item X locally from Sk and commits.

3. 2. 4. Update Transactions:

- When an update takes place at the primary servers of X, if Sj is not reachable from the primary servers then primary servers are not able to delete Xsec. However, Sj finds that the primary servers are not reachable from it and deletes the Xsec stored at Sj. This two way procedure of deleting Xsec helps in maintaining an up-to-date copy of Xsec

- Each of the primary servers also keep track of all the servers where secondary copy (replica) is placed. On the other hand, Servers with secondary copy keep record of their primary servers.

- When a transaction Tm that needs to update a data item X is originated at Sj, Tm requests an exclusive lock on all the servers in the group SGi where the primary copy of data item X resides. Tm then waits for a predetermined time $T_{timeout}$ to receive a response from all the servers in the primary group. If at least the majority (one more than half) of locks are obtained Tm continues; otherwise Tm is aborted.

  - Once the majority of locks are obtained, if Tm is a firm transaction, the nearest $\lfloor (NSGi / 2) + 1 \rfloor$ servers are updated.

  - If Tm is a soft transaction, the servers in the primary group are arranged in the decreasing order of their energy levels. The $\lfloor (NSGi / 2) + 1 \rfloor$ servers having the best

energy levels are then chosen for update. This helps in energy conservation at servers that have less energy level

- Two-way procedure to maintain / update copy is used.

    - If Sj cannot reach any of the primary server, then meanwhile a possible update could occur and since Sj is unreachable so won't get the update. This would bring inconsistent data available to users X, to address that Sj will delete $X_{sec.}$
    - If update takes place at primary server then all the secondary copies (which are reachable) will be deleted.

3. 2. 5. Replication:

- Replications plan uses a variation of Majority Voting Strategy [Draffan 1980] for read and update transactions.
    - Let a transaction T_m that READS/UPDATES X originates at a server S_j and let SGi be the Primary server group with respect to X. Let N_{SGi} be the number of servers in the group SGi. Now for UPDATES, any server Sj has to obtain at least $\lfloor (N_{SGi} / 2) + 1 \rfloor$ locks, whereas for READS Sj has to obtain at least $\lceil NSGi / 2 \rceil$ locks.

- Each primary server maintains update counters that keep track of the number of updates on the data item X. Similarly each secondary server S_j maintains a read counter that keeps track of the number of reads that are originated from Sj for the data item X. This read counter helps in deciding whether it is beneficial to allocate a replica of the data item X at the secondary server. i.e. if number of read request for a X is greater than the total no. of updates performed on X, then we place a secondary copy X_{sec} at the secondary server.

5. Comparison of GBRMAD and Quorum Based Technique:

Quorum based technique (QBS) provides a solution for data replication of data in mobile ad hoc networks, where partitions are more probable. The problem with QBS is that it will have in-consistent data. Consider an Update is sent to quorum Si, later another Update is sent for same data item to quorum Sj. Then all servers in the set Si-Sj will have old value of data item, hence cause to have inconsistency. QBS tries to minimize the difference Si-Sj to reduce in consistency. Another drawback is that QBS does not take Energy of nodes into considerations, which is primary consideration in GBRMAD technique.

GBRMAD Technique on the other hand, ensures the consistency of queried data, and energy conservation of mobile nodes, but at the same time its expensive in terms of processing time.

5. 1. Performance Issues of GBRMAD Technique and Proposed Improvements Scenarios:

5. 1. 1. No. of Groups

GBRMAD technique does not specify the value of *"m"* i.e. the selection of number of Groups. This parameter is of prime importance, because *m* determines how many groups are to be formed.

n = number of servers in the whole system.

k = number of data items.

m = number of groups.

With, $m < n < k$

If m is small:

For smaller values of m will result in formation of lesser groups. Since there are disjoint servers in groups so all the database will be divided into *m* parts and will be stored on the *k/m* servers. Resulting a heavy storage load on each server. For example if m=3 then each server will have to store $1/3^{rd}$ of the whole database at its local storage device. In an environment like MANET we do not have unlimited storage capacity, so it won't be feasible to have smaller value of *m*. That will increase the efficiency of local transaction in term of processing time but again will result in overhead for updates and storage.

If m is too large (i.e. $m \approx n$, since it cannot be more than n**)**

If number of server groups are nearly equal to the no. of servers itself, then each group could have at the most 1 or 2 servers assigned to them. In other words we will be storing disjoint data items on all servers. In GBRMAD technique, since we are assigning servers to groups and those servers will store data items, which are PRIMARY to that group which the servers belong to. Resulting in servers with *k/m* data items primary to them. In present technique if any of the server (as we have less servers in group) move out of the area or in the partition, then that particular data item is totally unavailable during that time for Read / Update transaction.

What should be value of "*m*":

Value of m should be such to distribute the load evenly over all the servers and to consider the communication cost involved for a single query, since there is an assumption

that all the servers have equal number of read accesses. Most importantly, when a server sends a request to read / update on data item x, it has to request read/update locks from all the members of the primary group of x, now if the servers in a group are two large then certainly locks required to read or write from that group will also be fairly high.

In order to distribute the database storage load and to reduce communication cost of requests for locks, the value of "*m*" is be chosen to be $O(\sqrt{n})$. This value of m, equally distribute the load among all the groups and since initially we have assumed that no. of read accesses are same for all servers, so all servers will have approximately equal no. of data items.

Resulting in $m < \sqrt{n} < n < k$

Simply, $m=O(n)$, so that the servers are divided equally into groups.

Example:

For a small database of size 45000 data items and given 36 servers. The Groups could be formed as

$$n = 36,\ m = 6,\ k = 45{,}000$$

then each group will have 12000/6 = 2000 data items belonging to it. Also each group will have 6 servers, with equal distribution of load. Additionally its beneficial to have medium size of groups specially in case of frequent network partitions.

5. 1. 2. Read Cost

- In GBRMAD, the major assumption is

$$\#\text{ of Reads} > \#\text{ of Updates}$$

So we have heavy majority of Read Accesses over Updates in the model used by GBRMAD, whereas the GBRMAD offers expensive (in terms of # of locks require) solution for simple MRV Read query (no matter Firm or Soft). The Data is consistent but the overhead is high for inter hosts communication, just to read a single data item, located remotely at one of the primary server. Simply the cost of Reading is fairly high and about the same as of Update.

To execute an update or a read transaction on data item X, originated at server S_k, then S_k has to request all the members of the primary group SGi of X (so corresponding cost $\approx N_{SGi}$ * single communication cost between two servers), additionally it has to wait for at least $\lfloor (N_{SGi} / 2) + 1 \rfloor$

locks, whereas for READS Sj has to obtain at least $\lceil N_{SGi} / 2 \rceil$ locks before it can read or update request on these servers.

It is clear that in this case the **Cost of Read ≈ Cost of Update.** Because, cost of update is just 1 more than required for read transaction. Therefore in an atmosphere where we have assumed that # of reads > # of writes, then its not feasible to keep the cost of read and update about the same. One solution to this problem is to determine the *ratios of read write transactions* and then use a corresponding *weighted voting* to perform a read or update transaction.

Simply, for Read transaction the communication cost is reduced as compare to the one proposed in GBRMAD. Now the point we will be compromising here is the cost of Updates which we have increased by the same factor which we are saving from Read transaction, but the advantage is that in our assumption the Read accesses are more frequent than Update accesses. SO it's a trade off between less operations which are expensive and more operations which are not-expensive (relatively).

Servers are assigned to groups based on their RRR (Read Request Ratios)

$$\mathbf{RR}_i = (\mathbf{A}_i \ / \ {}^{r=m}\Sigma_{r=1} \mathbf{A}_r)$$

Since we are assuming same Read Requests for all servers, so their Ratio is equal for now. In genera, the servers required by each group of a particular energy level is

$$N_{SGi}^{E_L} = RR_i \times N(E_L)$$

Where $N(E_L)$ is the number of servers belonging to any particular Energy Lavel class, as explained in GBRMAD Technique

Adaptive Weighted Majority Voting for Concurrency Control is proposed for the system. This technique differs from static behavior of classical MV scheme. In Mobile Ad-Hoc environment, we need to cut on the communication cost in-order to make system efficient and reduce the load involved for querying. Also in general the Replication is more suited for environments where the reads are more frequent than writes. Therefore, the ratios of read requests and write requests are a heuristic in our model. A weighted moving majority technique is used to reduce the cost of

Reads, on the other hand increasing cost of updates. Simply, by reducing the # of locks required to perform a read and increasing the locks required for update. Certainly ideal situation could be ROWA, but its not applicable in MANET, due to frequent partitions.

$$\text{\# of Locks for Read} = \left\lceil \frac{\text{Total \# of Write Requests in System}}{(\text{Total \# of Writes} + \text{Total \# of RR})} * 100 \right\rceil$$

$$\text{\# of Locks for Write} = \left\lceil \left[\frac{\text{Total \# of Read Requests in System}}{(\text{Total \# of Writes} + \text{Total \# of RR})} * 100 \right] + 1 \right\rceil$$

To avoid the convergence of this technique into pure ROWA, which indeed is not feasible due to expected probable partitions; we set a limit of the locks we can have starting from 50/50 to 20/80 yielding a resulting no. of locks for Reads to be between 20 – 50 and 51 - 81 for Writes.

Example:

For a database with total of 1200 transaction, if the Reads were 800 and Writes 400 then RLR is Required Locks for Read and RLW is Required (min.) Locks for Write.

***RLR** = # of Locks for a Read = Ceil [(400/1200)*100] = 34 % of total no. of servers*

and

***RLW**=# of Locks for Write = Floor [(800/1200)*100 +1] = 67 % of total no. of servers.*

Correctness of proposed Concurrency Control Scheme:

The proposed weighted majorities voting strategy effectively resolves the Concurrency Control Issue as well as reduce the cost on Read Request. The basic idea is simple: when a transaction needs a database tuple then it should not be altered by any other transaction meanwhile, it acquires a **lock** on the object. If an object is locked, then other transaction cannot change the value of that object. There are two modes of Locking, one is **Shared** and other is **Exclusive.** Simply, Write locks are Exclusive and Read locks are Shared i.e. there could be more than one read locks by different transactions but it is not possible to acquire the write lock once the tuple is read locked. For Writes we have to have an Exclusive lock on the tuple, which now cannot be read or written by any other transaction assuring the accuracy of data item.

In general the conflicts between transactions could be Read-Write and Write-Write. We can prove easily that once a transaction has acquired required # of exclusive locks on any data item, then no other transaction can perform read on that particular data item until the locks are released by the write transaction. Similarly if transaction T has acquire the desired number of locks for read, then other transactions can only read on that data item simultaneously, since Read lock is shared. Following Example will explain that our model sufficiently assures concurrency with the help of time stamp.

Example:

Consider the same # of transactions given in previous example on a data item X. Here we need the # of lock for Read =34 %. Let the number of servers in our system is 100.

Locks for Read = 34 & Locks for Write =67

For Write T if # of locks obtained are 67 then T can proceed and update the data item X. Since, the least number of locks required for a read are 34, which in this case not possible as 67 servers has already been locked on X, hence no Read-Write Conflict. At the same time no other transaction can get 67 locks from all server in total on X, therefore no Write-Write conflict. Now consider the accuracy of Writes and Reads.

If 67 of 100 Servers are updated on X then only these servers has updated value of X and rest of the servers have old or in-consistent value of X. But for a single read we need at least 34 locks and once these many locks are obtained in the worst case we would have 33 servers with old value on X and 1 with updated value. So we can choose the one with higher time stamp, resolving consistency issues.

5. 1. 3. Reduction of Communication Cost

In general the data items are fairly large than the size of the Time Stamp. So another proposed improvement in GBRMAD is when it requests the lock on a data item, position and Energy Level from primary servers, then at the same time we also request the Time Stamp of the remote copy of data item.

When received response, for MRV Transactions we select the server with highest time stamp among the desired # of locked servers and directly read from the ONE server with highest time stamp. This will save on communication cost and energy cost from additional data reads from all ½ locked servers like implemented in GBRMAD. We will save on communication as well as computation cost.

Experimental results show that this improvement has effectively reduced cost of read at any server. Since our basic objective is to reduce the cost of reads and at the same time make the query return ***Consistent Data*** at all times.

The part of Algorithm is presented below in Fig. 4 showing the proposed improvement by MGBRMAD from GBRMAD.

/* Modified Portion of Algorithm from GBRMAD */

Transaction T originating at server Sj (Primary w.r.t. X) Reads data item X

```
// Initialize a counter track the number of servers from which locks are obtained
Counter := 0;
// Initialize a Boolean value to false. If this Boolean is set to true a transaction is committed
SUCCESS := FALSE;

Identify and count the number of servers (N_SGi) belonging to the primary group of X
        // The following IF condition is executed if the server at which Tm originates is the primary
        // group of X and if Tm requires the most recently updated value.

IF (Sj belongs to the primary group of X) AND (T is MRV) THEN
        // Here the transactions request locks and relevant information from all the primary
        // Servers of the data item X

        FOR each server in the Primary Group (of X)

                Request a Lock & Time Stamp on X along with the position and energy level for each
                server in the primary group within the T_timeout period;
                ++Counter;

        END FOR;
        // Check if locks are obtained on Min. Required for READ on X. .

        IF Counter >= RLR ( Required(Min.) Locks for Read ) THEN

                // The servers are searched for highest time stamp on X and with min. distance
                // from Sj and also the one with highest energy level and time stamp.

                IF T is a FIRM THEN
                        Find the server with highest Time Stamp and at minimum distance from Sj;
                        // instead of sorting ,just searching so cost save

                ELSE IF T is a SOFT THEN
                        Find the server with highest Time Stamp and highest energy level;
                        // instead of sorting ,just searching so cost saving
                END IF;

                // Note the difference from GBRMAD Technique, which uses read from
                //about half the primary serves as shown in the statement above

                Read X from only Selected Primary Server
                SUCCESS:= TRUE;
        END IF

        // If the server at which T originates is from primary group of X
        // and if T is NORMAL (Does not require the most recently value of X)
```

```
        ELSE IF (Sj belongs to the primary group of X) AND (T is NORMAL) THEN

                IF T is ( FIRM)  OR  ( SOFT ) THEN
                        Request lock on its own database on X;
                END IF;

        //Check if the required lock is obtained. If so set success to true.
                IF required lock obtained
                        Read X;
                        SUCCESS:= TRUE;
                END IF;
        END IF;
//*************************************************************************

//If T originates from a secondary group Server
 // of  X  and if there is no replica of X at the secondary server.

        ELSE IF (Sj is not the Primary server w.r.t X) AND
                        (Replica of X, i.e., Xsec is absent at Sj ) THEN

                // Check if the Tm is MRV transaction

                IF (Tm is MRV) THEN

                        FOR each server in the primary group

                                Request  a lock & time stamp on X,
                                Energy Level and Position from the Primary Server
                                        within the Ttimeout period;
                                ++Counter;

                        END FOR;
                // Check if locks are obtained on Min. Required servers for READ on X.

                        IF Counter >=  RLR (Required(Min.) Locks for Read ) Then

                        // The servers are searched for highest time stamp on X
                        // and with min .distance/ from Sj and also the one with
                        // highest energy level and time stamp.

                IF T is FIRM THEN
                                        Find the server with highest
                                        Time Stamp and at minimum distance from Sj;

                        // instead of sorting ,just searching so cost save

                                ELSE IF T is SOFT THEN

                                        Find the server with highest
                        Time Stamp and highest energy level;
```

```
                    // instead of sorting ,just searching so cost save

                            END IF;

                    //The Replica Assignment is done if the number of reads is
                    // greater than the total # of  writes  on X i.e. Replica Feasible

                    IF (number of reads from Sj > Total number of writes on X) THEN
                            Obtain Xprim  from one Selected Primary Server

                            Allocate Xprim as Xsec at Sj;
                            Read Xsec;

                    ELSE
                            Read X from one Selected Primary Server;

                    END IF;
                    SUCCESS:  = TRUE

            END IF

// If T is a NORMAL execute the following section

ELSE
            IF T is FIRM THEN
                    Request lock on Nearest Primary Server;

            ELSE IF T is SOFT THEN

                    Request lock on Primary Server with Highest Energy Level;

            END IF;

            IF required lock obtained
                    Read X;
                    SUCCESS:=TRUE;
            END IF;
END IF;

//If T originates from a secondary group Server
 // of  X  and if there is replica present of X
            ELSE IF (Sj belongs to the secondary group w.r.t X) AND (replica of X, i.e., Xsec is present
            at Sj) THEN
                    IF T is MRV
                            Check for the Validity Flag of Data (Xsec)
                            IF VALID
                                    Request lock for Xsec on Sj ;
                            ELSE
                                    Repeat the Process for MRV Read Transaction from Primary Servers
                            ENDIF
                    ELSE IF T is NORMAL
                                    Request lock for Xsec on Sj;
ENDIF
```

```
                    //Check if the required lock is obtained. If so set success to true.
          IF required lock obtained
                    Read Xsec;
                    SUCCESS:=TRUE;
          END IF;
END IF;
// If success is true, the transaction can be committed else it is aborted

IF (SUCCESS) THEN
          COMMIT T;
ELSE
          ABORT T;
END IF;

// If a deadline of the committed firm transaction is missed compensate the transaction
IF (T  is COMMITTED ) AND ( DEADLINE  is missed for FIRM T) THEN
          COMPENSATE T;
END IF;
```

Figure 4: Modified GBRMAD Execution Algorithm

Deletion of Replicas:

If a Secondary copy of data item X is assigned to server Sj, after a time interval (depending on the server's mobility pattern) if Sj is unreachable (due to possible network partition) by the primary group members of Data item X, then for GBRMAD it has to be deleted from all secondary servers to save the cost on Replica updates. When the secondary servers are un-reachable then secondary servers preemptively implement it after a certain time interval T_{expire} provided the primary servers are not reachable. The value of T_{expire} is obtained from the history of update frequencies of that data item after which it has to check for the reach-ability of the primary hosts. In GBRMAD we simply delete all the replicas, which in case of partition will make that data item totally unavailable in that partition even for the Normal Transactions.

An improved solution is provided which simply mark the replica of data item X on a secondary server Sj un-reliable or expired after time interval T_{expire} provided the secondary server is unreachable from primary servers. A Transaction T will be checked if it's a MRV /Normal Transaction. If the Transaction is Normal so we could still server it without compromising on accuracy requirement for MRV transactions, which will first check the flag for expiry of that replica of X.

However by not deleting and simply *marking* it obsolete or out-dated in the event when primary servers of X are un-reachable due to some partitions, we could save on the cost involved in serving Normal transaction for data item X originated at Server Sj. For MRV transaction we have to wait to get most recent value, if not available then transaction will be aborted.

6. **Simulation Model:**

Simulation is the use of a model to develop conclusions that provide insight on the behavior of any real world elements.

Major types of Simulation includes

➔ Monte Carlo Simulation

A scheme employing random numbers that is used for solving certain stochasticor deterministic problem where the passage of time plays no role.

➔ Continous Simulation

Continous simulation is concerned with the modeling a set oequations that represent a system over time. This system may consist of algebric, differencial or difference equations set up to change continosly with tim.

➔ Discreet Event Simulation

Characterized by the passage of blocks of time during which nothing happens and in punctuated by events that change the state of the system.

Visual SLAM (*Simulation Language*) in ***Awesim*** is used for our simulation. It allows systems having process-oriented, event-oriented and continuous components to be integrated in a single system model. To measure performance of this proposed technique we constructed a model and different experiments and test were performed for performance analysis of GBRMAD and comparing its performance with the proposed MGBRMAD (Modified GBRMAD).

As a first step for simulation following is the System Definition for GBRMAD as well as for MGBRMAD, suggesting methods for data replication in Mobile Ad-hoc Network Environment. To ensure reliability of obtained information, improve processing time for Energy and Time critical mobile ad-hoc environments.

Major Goals

- ➔ Decrease Energy Consumption
- ➔ Achieve Accuracy of Queries.
- ➔ Time for transaction should be reasonably compareable with existing non-energy critical techniques.

Assumptions

- ➔ All LMHs have same (fairly large) storage capacity.
- ➔ Transaction could be Critical / Non Critical.

- ➔ Transaction could be Normal / MRV(Most Recent Value)
- ➔ No. of Reads for all LMHs are same.
- ➔ All transactions are compansable
- ➔ # of Reads >> # of Writes or updates
- ➔ Initial static allocation of data.

Terms:

GROUP ➔ Collection of servers (LMHs), such that all groups are mutually exclusive, and together define whole set of servers. Groups are Primary and Secondary w.r.t. a particular data item.

Primary Copy ➔ Primary copy of data stored at their primary servers (in its group)

Secondry Copy ➔ Replica of primary copy at one of the secondry server.

Known Items / Variables (assuming from previous history of manet and transactions)

- ➔ m = No. of Server Groups
- ➔ n = Total no. of servers in entire system (n > m)
- ➔ k = total no of data items in entire database.
- ➔ J = No. of different Enrgy Levels
- ➔ Energy levels of all servers (stored in an array)
- ➔ No. of read accesses for all groups (stored in a global array)
- ➔ Array, having # of servers of all energy level.

Initial Model of Simulation:

Transaction:

All transactions are defined as ***Entities*** with the following attributes.

1. Creation Time
2. Transaction ID
3. Transaction is Update / Read (0 Read, 1 Write)
4. Data item to be transacted (say its data item 13)
5. Deadline ((Estimated Run Time * Slack ratio) +communication cost)
6. Assigned Server
7. Type I (Firm 0, Soft 1)
8. Type II (MRV 0, Normal 1)

9. ID of the SMH, which has initiated the transaction.
10. Final Status (Abort / Commit)

Large Mobile Hosts:

The Mobile Hosts are defined as ***Resources*** in the model. Followings are the associated attributes.

1. Resource ID
2. Group No.
3. X Position
4. Y Position
5. Energy Level
6. Number of Successful Transactions
7. Number of Aborted Transactions
8. Total time in Doze Mode
9. Total time in Active mode.
10. Number of Read Transactions Handled.
11. Number of Write Transactions Handled.
12. Each server's Primary Data items with time stamps.
13. Each server's Secondary Data items (if any) with time stamp(s).

Globally Available Items:

1. Data Items, their time stamps, corresponding Groups, and Servers
2. Position of Each Server (X, Y co-ordinates in 1000 x 1000 matrix)
3. Current Energy Level of each Server.
4. Total No. of Successful Read Accesses in System
5. Total No. of Aborted Read Transactions
6. Total No. of Successful Update transaction
7. Total No. of Aborted Update Transaction

6. 1. Description of Models:

All the resources of the system are initialized in the beginning. The position of each resource is obtained from random distribution of X & Y co-ordinates. Initially, assumed a system composed of 9 LMHs and 45 SMHs. (as assumed the SMHs are moving along with servers i.e. they are statically attached).

- Initially all the 9 servers are distributed randomly in a square region of 1000 x 1000 grid, i.e. 1000 length units each side.
- After a certain time defined as residence latency, LMHs will move randomly from its current position to not more than 150 units in any random direction. This residence latency was one of the heuristic in our model.
- A wireless link is assumed to be present between a pair of nodes if they are within Radius of influence (in units of length) of each other.
- No. of groups "m" is assigned an initial value of Square root of total # of servers i.e. 3.
- The # of data items are 27.
- Time between successive queries by SMHs is exponentially distributed with a mean time of Dead Line. This is also one of the heuristic for performance analysis of GBRMAD and MGBRMAD. The slack ratio is varied for different experiments.

$$T_{Timeout} = ((2 * \text{Time Communication cost}) + \text{Time required to wait for a lock})$$

$$\textbf{Dead Line} = \{\text{Time. Generation} + (\text{Runtime Estimate} * \text{Slack Ratio}) + (2 * \text{Communication Cost})\}$$

Fig. 4, 5,6 gives the pictorial flow of transaction in GBRMAD.

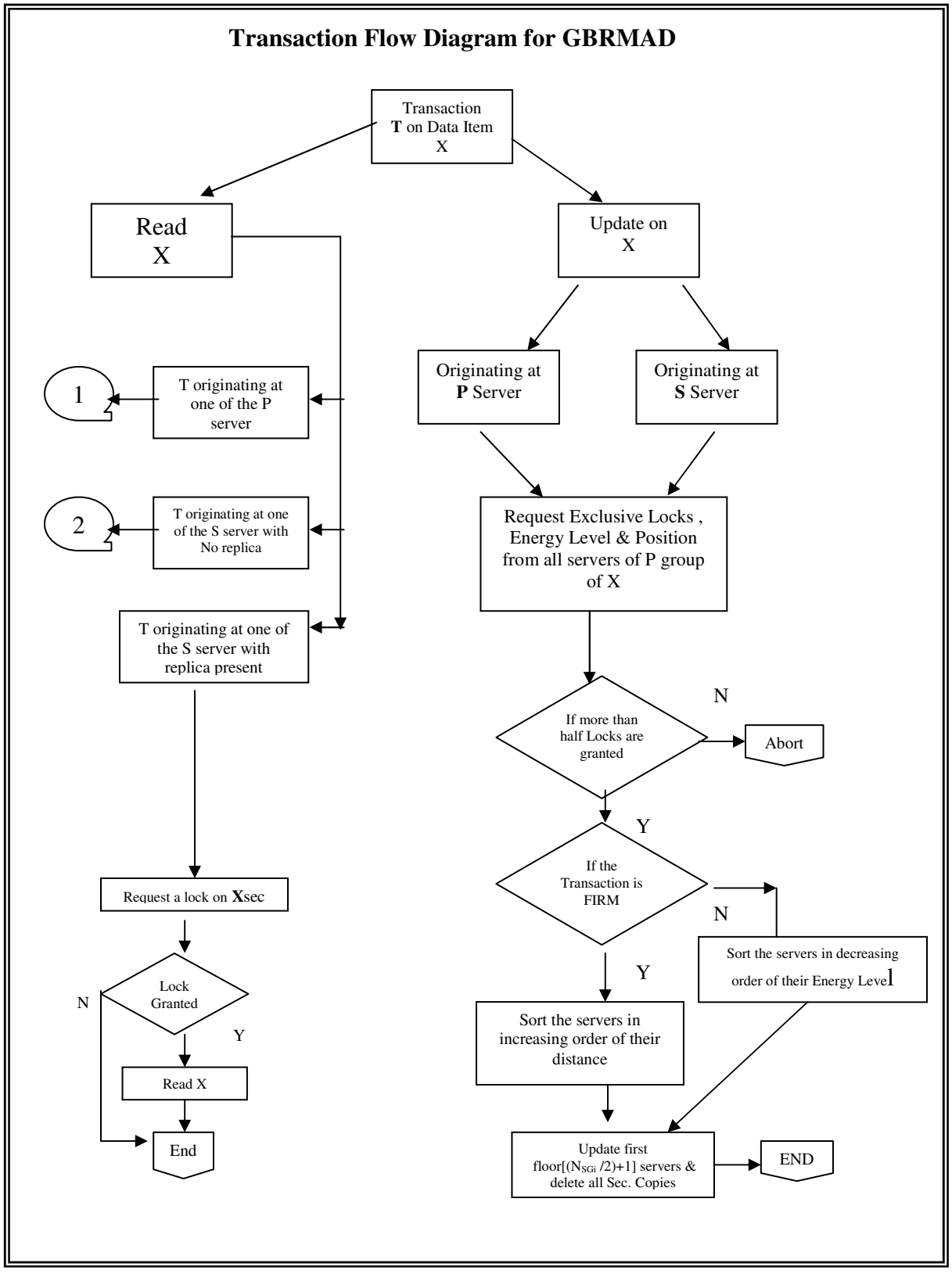
Transaction Flow Diagram for GBRMAD
Transaction T on Data Item X
Read X
Update on X
1
T originating at one of the P server
2
T originating at one of the S server with No replica
T originating at one of the S server with replica present
Request a lock on Xsec
Lock Granted
N
Y
Read X
End
Originating at P Server
Originating at S Server
Request Exclusive Locks , Energy Level & Position from all servers of P group of X
If more than half Locks are granted
N
Abort
Y
If the Transaction is FIRM
N
Sort the servers in decreasing order of their Energy Level
Y
Sort the servers in increasing order of their distance
Update first floor[(NSGi /2)+1] servers & delete all Sec. Copies
END

FIG. 5

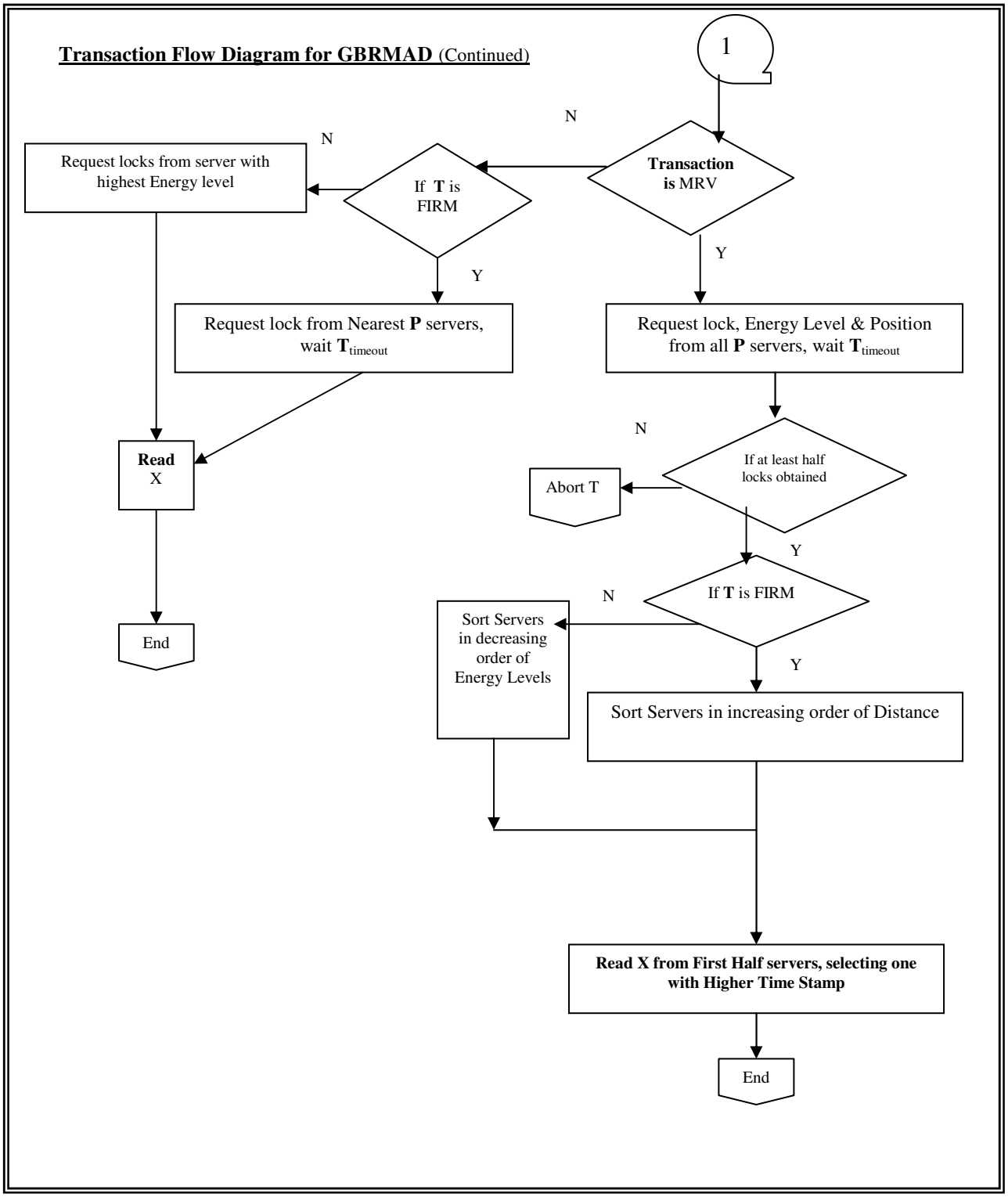
Transaction Flow Diagram for GBRMAD (Continued)
1
N
Transaction is MRV
Y
If T is FIRM
N
Request locks from server with highest Energy level
Y
Request lock from Nearest P servers, wait $T_{timeout}$
Request lock, Energy Level & Position from all P servers, wait $T_{timeout}$
N
If at least half locks obtained
Abort T
Y
If T is FIRM
N
Sort Servers in decreasing order of Energy Levels
Y
Sort Servers in increasing order of Distance
Read X
End
Read X from First Half servers, selecting one with Higher Time Stamp
End

FIG. 6

Transaction Flow Diagram for GBRMAD (Continued)

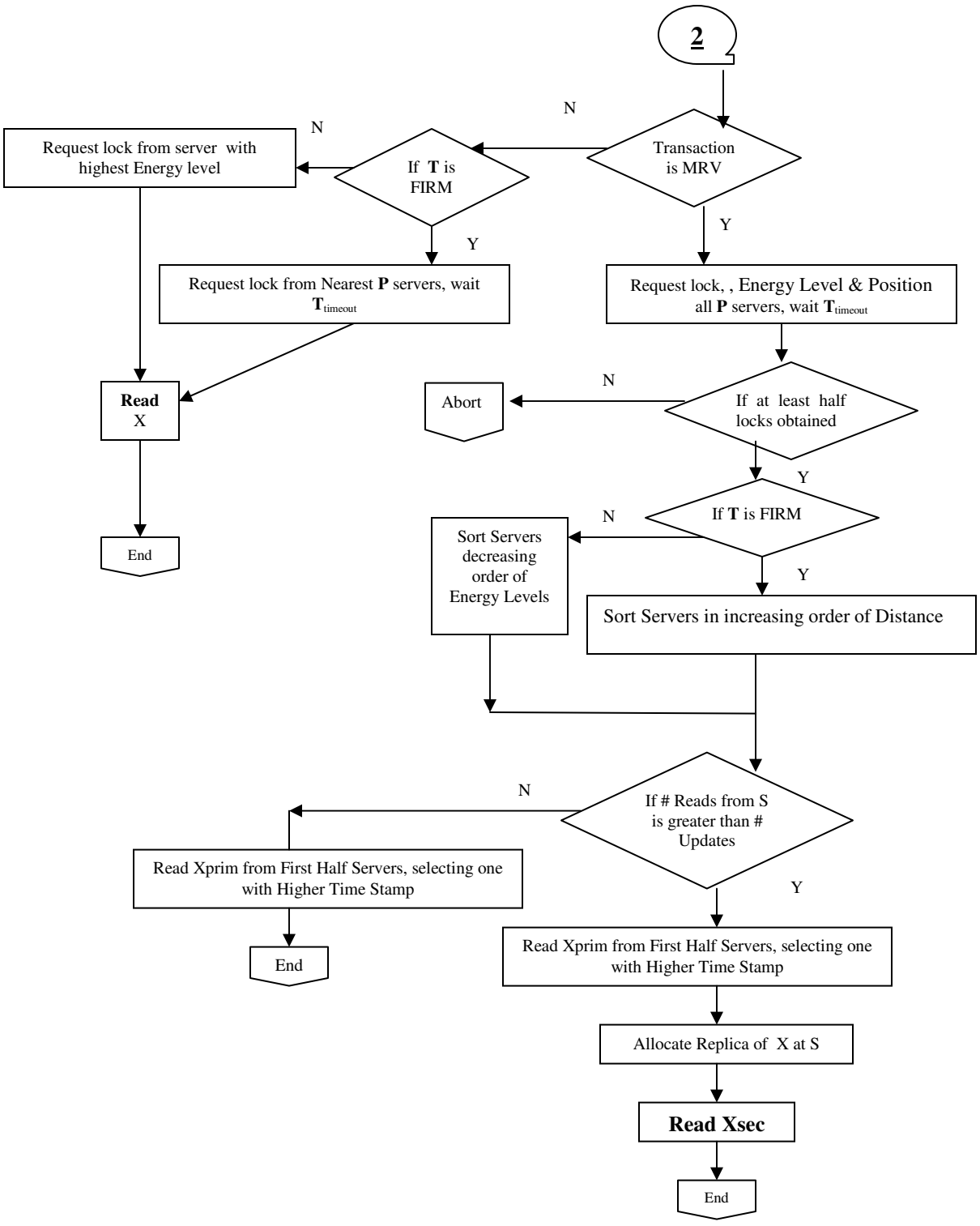

FIG. 7

6. 2. Parameters of Static and Dynamic nature:

The bandwidth of the wireless medium is chosen to be 1 Mbps between the LMHs, where as bandwidth assumed to be 100 kbps between LMHs and SMHs. The CPU power, memory access time and word size are chosen based on the DEC 3000 model machine [DEC, 1993] for LMH. We have assumed the main memory is large enough to store the entire the database, this assumption eliminates disk access time and other parameters required, regarding the spin issues of Hard Disks. The number of operations per site transactions is taken to be uniformly distributed between 2 to 10. The slack ratio is used to assign deadlines for each transaction. The power dissipation rate of an SMH is taken from [Imielinski, 1994] considering the SMH to be a palmtop with 386 processor, 20MHz and 4MB RAM. The power dissipation rate of an LMH is taken from [Michelle, 1996] as 170W per hour. Cost of Sorting is considered to be optimal (i.e. s Log_2 s) for sorting *s* servers. Time units are considered as milliseconds in Simulation.

Table 1: Static Parameters of Simulation Model

Parameter	Representing	Set Value
Bandwidth 1	Bandwidth of wireless medium LMH-LMH	*1 Mbps*
Bandwidth 2	Bandwidth of wireless medium SMH-LMH	*100 kbps*
CPU_power_LMH	CPU Power of LMH	*140 MIPS*
CPU_power_SMH	CPU Power of SMH	*4 MIPS*
Energy_Levels	Energy Levels of Servers (J levels)	*1000-800-600*
LMH_power_in_Active	LMH Power Dissipation Rate in active mode	*170 Watts*
LMH_power_Doze	LMH Power Dissip. Rate in Doze Mode	*20 Watts*
Mem_access_time	Main Memory access time per word	*0.00018 ms*
Mobility_LMH	Movement of LMHs after time period	*100 unit*
Num_SMHs	No. of Small Mobile Hosts	*45*
Num_of_Class	Number of Energy Level Classes	*3*
Prob_Write	Probability of Write operations	*0.2*
Prob_read	Probability of read operations	*0.5*

Num_Groups	No. of Groups	3
SMH_power_Active	SMH Power Dissipation Rate in Active	*7 Watts*
SMH_Power_Doze	Power Dissip. In Doze Mode of SMHs	*1 Watts*
Size_Data_Item	Size of Data Item Requested for Query / Update	*1 kB*
Word_size	Number of bytes per word	*4*
Num_LMHs	No. of Large Mobile Hosts	9
Num_Data_Items	Number of Data Items	27
Num_Data_per_Server	Number of Data items primary per Server	9
Pre_op	Preprocess one operation	0.000007 ms
Pre_trans	Preprocess one transaction	0.0072 ms
Et_tran	End transaction	0.0054 ms
$T_{timeout}$	Waiting Period for response	0.0007 mSec

Table 2: Dynamic Parameters of Simulation Model

Parameter	Meaning	Default	Range
IAT	Inter Arrival Time b/w transactions	EXPON (0.2)	EXPON(0.2 to 1.0)
Radius_LMH	Radius of influence of LMH	*200*	200 - 800
Prob_FIRM	Probability that a transaction is Firm	0.5	0.1 - 1
Prob_SOFT	Probability that a transaction is Soft	0.5	0.1 – 1
Prob_NORMAL	Probability that a transaction is Normal	0.5	0.1 - 1

Prob_MRV	Probably that a Transaction is MRV	0.5	0.1 – 1
Pob_FIRM_MRV	Probability that a transaction is FIRM and MRV	0.25	0.01 - 1
Prob_FIRM_NORMAL	Probability that a transaction is FIRM and NORMAL	0.25	0.01 – 1
Prob_SOFT_MRV	Probability that a transaction is SOFT and MRV	0.25	0.01 – 1
Prob_SOFT_NORMAL	Probability that a Transaction is SOFT and NORMAL	0.25	0.01 – 1
Slack_ratio	Slack ratio to calculate deadline	20	10-30

6. 3. Performance Metrics:

In our simulation, the performance metrics for evaluating the performance of GBRMAD technique and MGBRMAD are as follows:

- Energy Consumption of the Entire System.
- Average Transaction Execution Time.
- Avg. Execution time for each type of Transactions
    - Firm & MRV
    - Firm & Normal
    - Soft & MRV
    - Soft & Normal.

  and impact on % age of Transaction missing deadline
- %age of Transactions missing Deadline with host reach-ability

Energy Consumption of LMH = {Initial Energy of LMH - ((Time spent in Active mode * Active mode power consumption rate) + (Time spent in doze mode * Doze mode power consumption rate))}

Energy Consumption of Entire System = **Σ** Energy Consumption of LMH

Avg. Transaction Execution Time = (**Σ** Transaction Execution time) / Total no. of Transactions

$$\textbf{\% of Transaction Missing Deadline} = \left\{ \frac{\#of\_Transactions\_missed\_deadline}{\#of\_Transactions\_processed} \right\} * 100$$

$$\textbf{\% of Transaction Committed} = \left\{ \frac{\#of\_Transactio\ ns\_Commited}{\#of\_Transactio\ ns\_processed} \right\} * 100$$

7. Experimental Results:

7. 1. Effect of Host Reach-ability on Transaction Abort Rate:

Radius of influence is a direct measure of host reach-ability. Fig. 8 depicts the effect of increasing radius of influence on %age of Aborted Transactions. Since Radius of Influence in our model plays major role in terms of Partition, smaller values of Radius of influence means that there will be more network partitions, resulting more Transactions missing deadline. When a server is un-reachable then its more likely that transaction can miss deadline, even for Soft Transactions, which has larger deadlines. The Graph shows the two different deadlines for different slack ratios. There is not a drastic change with deadlines, since the major contribution is coming from the Soft and MRV Transactions.

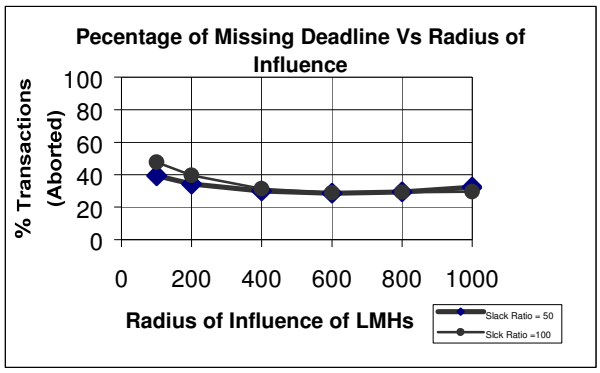

FIG. 8: Effect of radius of influence on %age of Missing Deadline

7. 2. Probability of MRV Transactions & its effect on Transaction Abort Rate:

The different probability of MRV Transactions has a pronounced impact on percentage of Aborted Transactions. With varying probability of Most Recent Value (MRV) Transactions there is considerable change in the system's performance. Since the MRV Transactions need most updated value, hence higher cost is involved in term of inter-LMHs communication. Therefore increasing the probability of missing a deadline, especially in the cases when servers are busy serving other clients. In that case, the requests have to wait in the waiting queue of LMHs, which could more degradation on the part of successful transactions. Fig. 9 has two different curves, which exhibits nearly same behavior, with the changing probabilities of MRV Transaction. The Upper curve show the effect of more FIRM (Prob. 0.75) Transaction in the system whereas, the lower one is with less FIRM (Prob. 0.25) Transaction. The behavior is self explanatory, as for majority of FIRM Transaction, more probable to miss the deadline. The lower bound shows the effect of partitioning. The value of radius was taken medium i.e. 500 units for 1000 transaction.

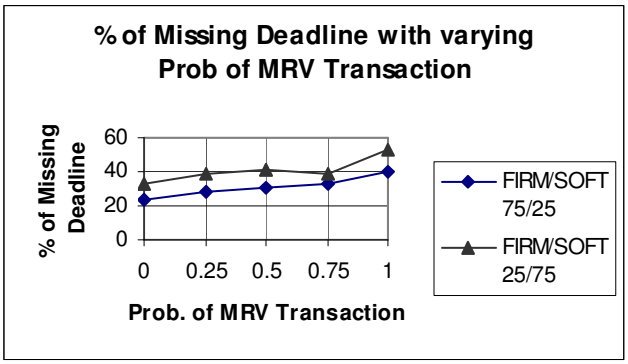

FIG. 9: Effect of Prob. of Transaction as MRV on %age Missing Deadline

7. 3. Percentage of Aborted Transaction with varying Probability of FIRM Transaction:

Fig. 10 explains the impact on percentage of transaction missing deadline, from varying probability of FIRM Transactions. Here the proportion of MRV and Normal transaction were 50/50 i.e. equally likely. Experimental results are a good match with theoretical anticipations. By increasing probability of FIRM transaction, there will be more transaction with tight deadline and hence resources are limited so more chance to miss deadline.

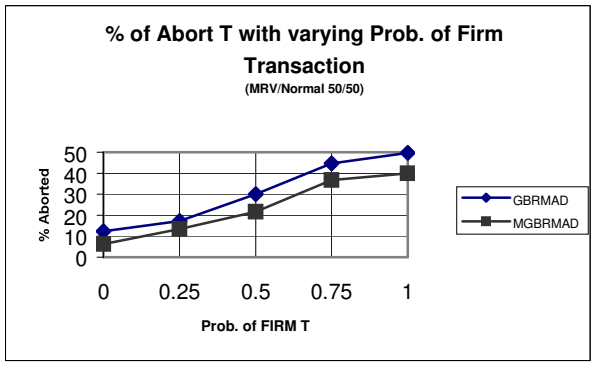

FIG. 10: Effect Prob. of FIRM Transactions on %age Missing Deadline

7. 4. Energy Consumption of LMHs:

Total Energy Consumption of individual LMHs from 1000 transaction are shown in the graph below. During the experiment, the probability of Firm / Soft Transactions was equal, i.e. .5 /.5 additionally the probability of MRV / Normal Transaction was also same. The interesting graph in Fig. 11 gives the energy consumption of LMHs in watts. The MGBRMAD has outperformed the GBRMAD in terms of Energy consumption, since the former costs almost 1/3rd less for the MRV-FIRM transactions than the later. But the FIRM-MRV transaction are only 25 %, therefore the save in energy is not very drastic in this graph.

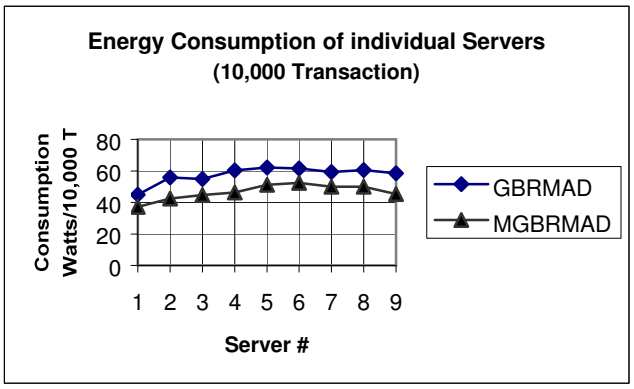

FIG. 11: Energy Consumption of individual severs

7. 5. Transaction Execution Time:

There are different types of transactions in our system and the average execution time defines an overall system performance in terms of time. Fig. 12 shows that MGBRMAD has an edge over the GBRMAD, mostly due to its cost-efficient behavior for Normal Transaction. Since the MGBRMAD does not delete replicas unlike GBRMAD, rather it just marks them un-reliable after the data item's update latency period, hence making the replicated copy still available for all Normal Transactions, which in overall are 50% of the whole transaction. No matter if its FIRM or Soft both types could be served effectively using replicas.

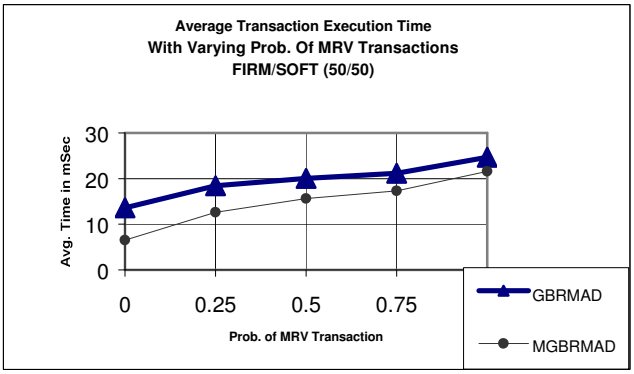

FIG. 12: Average Transaction Time with varying Prob. of MRV Transactions

7. 6. Effect of MRV Transactions on Energy Consumption of the system:

The overall energy consumption of the system is the energy consumed by all LMHs and it is affected by the percentage of MRV transactions in the system. The behavior is about the same what was anticipated theoretically. Fig. 13 gives a comparison between energy consumption of LMHs for both GBRMAD and MGBRMAD techniques over the varying values of MRV Transactions. Since the later is improved version of the first so, it saves energy mainly where it requires lesser locks (weighted majority) for read transaction, but the significant change is for Normal transactions where the GBRMAD tends to cost

drastically more, mainly because of deletion of replicas which are conserved in MGBRMAD making them available for Normal transactions.

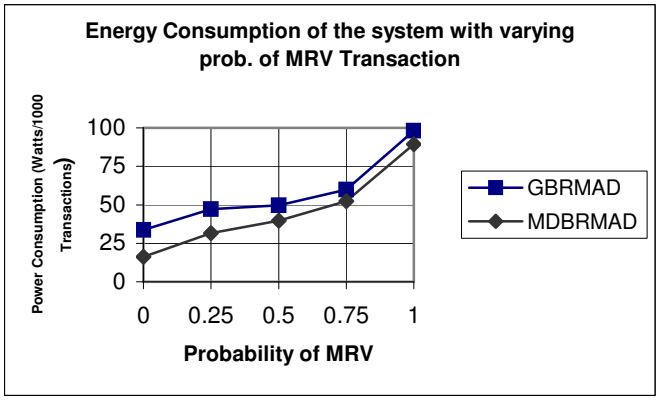

FIG. 13: Energy Consumption of the System with varying prob. of MRV Transactions

7. 7. Impact of Transaction Inter-arrival rate on Transaction Abort Rate:

This experiment shows the effect of inter-arrival delay between successive transactions on overall %age of aborted transactions. The difference in performance of both the techniques is not very significantly different, mostly because the inter-arrival time directly increases the no. of transactions present in the system at any instance of time. The inter-arrival time was taken as exponential function with mean values shown as x-axis of the graph, with 1000 transactions in total. For smaller values of inter-arrival time more transaction are aborted where as with increasing delay between arrival of two successive transaction, the no. of transactions aborted decreases significantly. Here the residual aborted transactions reflect the presence of probable partitions in the system. In general MGBRMAD has better performance in this case also.

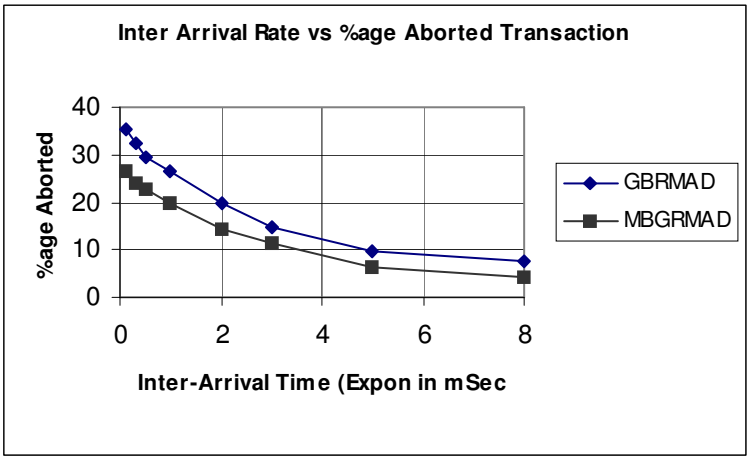

FIG. 14: Impact of Transaction Inter-Arrival Rate on %age Aborted Transactions

7. 8. Effect of Transaction Inter-arrival delay on Energy Consumption:

Energy Consumption of the system has interesting relation with varying inter-arrival time between successive transactions. In Fig.15, the lower line shows the number of aborted transactions simultaneously. Its obvious that with more transaction in the system, more transaction will be missing deadlines or the corresponding primary servers could be busy serving other transaction, resulting in lesser overall energy consumption of the system. With increasing delay between arrivals of transactions, there will be fewer transaction aborting and hence resulting more work done by servers and eventually more Energy Cost. In our model, the MGBRMAD saves a little more energy as compare to GBRMAD, due to higher availability of data for Normal transaction, fewer locks needed for reads and data acquisition from only one primary server per transaction. All the factors make MGBRMAD better choice over GBRMAD, especially it performs drastically better in the case of larger size of data items.

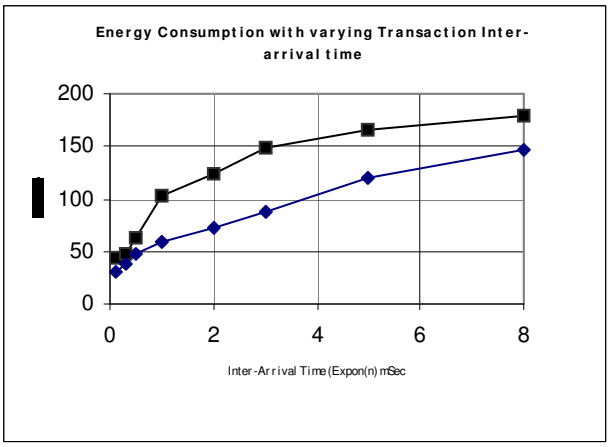

FIG. 15: Energy Consumption of the System with varying Transaction Inter-Arrival Time

7. 9. Effect of varying proportion of Read Transactions:

Since GBRMAD costs almost equal for Read / Write transactions, so there will not be much difference in the overall system performance. Whereas, the MGBRMAD technique uses ratio between read and write transactions and adaptively calculate the number of minimum locks required for Reads or Writes, which for worst case becomes nearly equal in performance to GBRMAD where it will be majority of Update transaction in the system. On the other hand MGBRMAD outperforms GBRMAD when there are more Read Transaction in the system, since GBRMAD behaves statically irrespective of the ratio of Read/Write. The difference is shown in Fig.16.

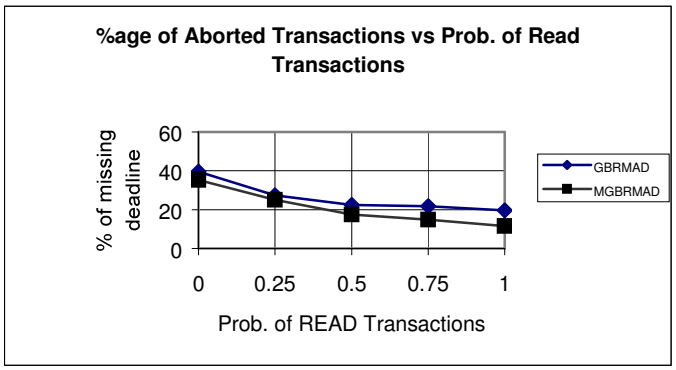

FIG. 16: Impact of Prob. of Read Transactions on Transaction Inter-Arrival Rate on %age Aborted Transactions

7. 10. Effect of Residence Latency on Transaction Abort rate:

Fig. 17 shows the effect of Residence Latency on percentage of transactions missing deadline. The comparison between GBRMAD and MGBRMAD shows that there is significant improvement in terms of no. of successful transactions. Residence latency was defined as average time spent by a mobile host at a certain position. i.e. the Mobile Host will change its position after a certain interval of time. Since the transactions occur with in units of milliseconds, so in real life model, its unlikely that hosts can move noticeable distance (in terms of their reach ability) during that small interval of time. The results achieved are with slight fluctuations, depends upon the time (in mSec) spent at a certain location, so initially for smaller values of latency period, the hosts are more mobile, so more mobility, hence more probable network partitions, although for smaller interval of time. For larger values of latency period, the no. of transactions missing deadline are relatively small, since LMHs stay at a point for longer interval of time, therefore for unreachable hosts the servers will be unreachable for longer period of time. Overall MGBRMAD and GBRMAD exhibit similar behavior with changing latency of residence.

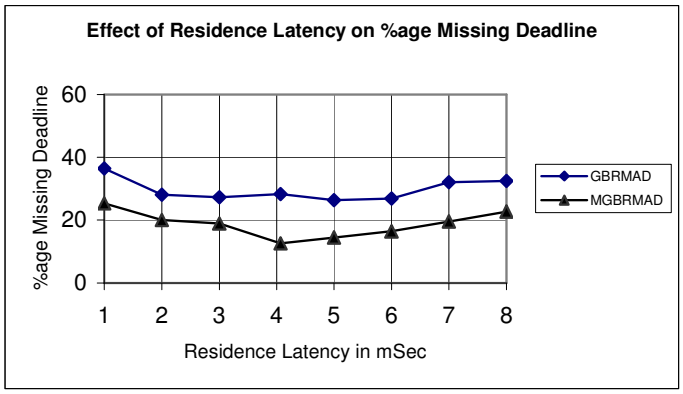

FIG. 17: Effect of Residence Latency of LMHs on %age Aborted Transaction

7. 11. Energy Consumption of the System with varying Prob. of Read Transactions:

With increasing number of Update or Write Transactions (in general, update is always more expensive in distributed databases) the Energy Consumption is also increased in our model, justification is that when there are more Updates, more servers are involved and hence more waiting time, and when server has some job in the waiting queue it cannot go into doze mode, therefore the energy consumption is increased. Also in MGBRMAD technique, which in worst case performs as GBRMAD and outperforms in overall energy consumption of the system, again because of adaptive nature of lock management.

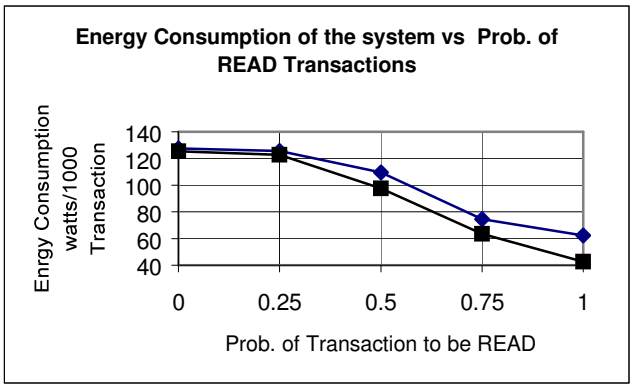

FIG. 18: Energy Consumption of System with varying Prob. of Read Transactions

8. Conclusion & Prospects:

Theoretical analysis performed by [Gruenwald, 2000][I] to compare performance of QBS and GBRMAD technique reflects higher cost in term of time for GBRMAD than for QBS, i.e. QBS performs better in time but on the energy side it does not consider the energy limitations at all, so consumes more energy than GBRMAD. Important point of compromise is between the higher cost and data consistency, which could be vital in military environment, where consistency of data is required. The major difference between the QBS and GBRMAD is the degree of application; the QBS is situation specific, i.e. it is suitable for some situations like fire fighting or other temporary disaster recoveries. QBS lacks any Concurrency control mechanism and address limited issues in MANET databases. Whereas GBRMAD being among first efforts in MANET environment provides data replication along with assured consistency, gives a comprehensive solution addressing major issues including concurrency control, data consistency and most important Power Awareness. It saves energy by distributing load among the servers with high power level (where possible like Soft Transaction) and by making use of electronics advancements for utilizing DOZE and ACTIVE modes.

The authors have tried to address areas of GBRMAD where some significant improvements were needed. MGBRMAD, basically uses the same model and structure of GBRMAD, but enhance the performance using adaptive nature of required locks assignments, replica availability, instead of deletion especially for normal transactions and finally requirement of single primary server for data acquisition, unlike from majority of the servers which of course is very expensive and crucial.

Experimental results for Energy consumption and Transaction execution and abort rate shows that MGBRMAD performs better than GBRMAD. In overall picture it out performs GBRMAD, where as in specific cases where for example larger data size is involved, MGBRMAD is far more efficient than its predecessor. For systems with higher ratios of reads over updates and with all kind of transactions present with equal probabilities (MRV, NORMAL, FIRM, SOFT), MGBRMAD performs approx. 25-30% better than GBRMAD. Although, these modifications (MGBRMAD) have significant effect on system's performance, but there are few areas like failure and recovery management, which has to be considered in future. We will be performing more experiments to study MGBRMAD's performance and the effect of different number and mobility of groups, which was not performed extensively due to model limitations. Finally, MGBRMAD gives an improved solution for data replication in a frequently partition-able environment like Mobile Ad hoc networks.

9. Reference:

[Dirckze, 1999] Dirckze R. A., "*Transaction Management in Mobile Multidatabases"*, Ph. D. Dissertation, Department of Computer Science, University of Oklahoma, Norman, 1999.

[Draffan, 1980] Draffan I. W., Poole F., "Distributed Databases", Cambridge University Press, Cambridge, pages 226-228, 1980.

[Gruenwald, 2000][I] Gruenwald, Sudeep Mohile, *"Energy Efficient Data replication for MANET"*, A study report , Univ. of Oklahoma 2000.

[Gruenwald, 2000][II} Gruenwald L., *"An Energy-Efficient Transaction Management Technique For Real-Time Mobile Database Systems in Ad Hoc Network Environments",* NSF Proposal, 2000.

[Huang, 1994] Huang, Y., Sistla, P., Wolfson, O., "*Data Replication for Mobile Computers."* Vol 5, pages 13-24, SIGMOD 1994.

[Imielinski, 1994] Imielinski T., Viswanathan S., "Adaptive Wireless Information Systems", in Proceedings of SIGDBS (Special Interest Group in Database Systems) Conference, October 1994

[Karumanchi, 1999] Karumanchi G., Muralidharan S., Prakash R., "*Information Dissemination in Partitionable Mobile Ad Hoc Networks"* Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems, 18-21 October, 1999, Lausanne, Switzerland.

[Lee 1998] Lee K. K. S. Chin, Y. H. *" A new Replication strategy for Unforeseeable Disconnection under Agent-Based Mobile Computing System*" IEEE Proceedings of the International Conference on Parallel and Distributed Systems, 1998.

[Mohan 1994] Mohan S., Jain R., "*Two user location strategies for personal communications services"*, IEEE Personal Communications, pages 42-50, First Quarter 1994.

[Shivakumar, 1997] Shivakumar N., Jannink J., Widom J., "*Per-user Profile Replication in Mobile Environments: Algorithms, Analysis, and Simulation Results*" Mobile Networks and Applications, pages 129-140, vol. 2, 1997.

[Wu, 1998] Wu, S., Chang, Y., "*An Active Replication Scheme for Mobile Data Management*." IEEE proceedings of 6th International Conference on Database Systems for Advanced Applications, 1998

[Zalslavsky, 1996] Zaslavsky, A., Faiz, M., Srinivasan, B., Rasheed, A., Lai, S.J., "Primary Copy Method and its Modifications for Database Replication in Distributed Mobile Computing Environment", 15th IEEE International Symposium on Reliable Distributed Computing Systems, Canada, October 1996.