Distrib Parallel Databases (2007) 22:27–54
DOI 10.1007/s10619-006-7008-2

Managing real-time database transactions in mobile ad-hoc networks

Le Gruenwald · Shankar M. Banik · Chuo N. Lau

Published online: 20 January 2007

Abstract In a mobile ad-hoc network (MANET), mobile hosts can move freely and communicate with each other directly through a wireless medium without the existence of a fixed wired infrastructure. MANET is typically used in battlefields and disaster recovery situations where it is not feasible to have a fixed network. Techniques that manage database transactions in MANET need to address additional issues such as host mobility, energy limitation and real-time constraints. This paper proposes a solution for transaction management that reduces the number of transactions missing deadlines while balancing the energy consumption by the mobile hosts in the system. This paper then reports the simulation experiments that were conducted to evaluate the performance of the proposed solution in terms of number of transactions missing deadlines, total energy consumption and the distribution of energy consumption among mobile hosts.

Keywords Mobile database · Real-time database · Transaction management · Ad-hoc networks

1 Introduction

Rapid developments in wireless technology have enabled mobile users to access data from different sites. The Mobile MultiDatabase Management Systems (MMDBMSs) ensure convenient and efficient access of databases for the mobile users. Transaction Manager (TM) is a vital component that provides reliable and consistent units of computing to its users in MMDBMSs. The wireless communication medium poses

Recommended by: Ahmed Elmagarmid

This work is partially supported by the National Science Foundation grants No. EIA-9973465 and IIS-0312746.

L. Gruenwald (✉) · S. M. Banik · C. N. Lau
The University of Oklahoma, School of Computer Science, Norman, OK 73019, USA
e-mail: ggruenwald@ou.edu

two new issues that need to be taken care of by the TM, i.e., frequent disconnection and migration. Disconnection in wireless systems cannot be treated as catastrophic failures that result in aborted transactions. When a disconnection occurs, TM needs to determine the status of the user, and if reconnection is expected, the transaction must not be aborted. However, even if reconnection is not expected, aborting a transaction should be postponed as long as possible since the status of the user can only be predicted. Also, a disconnected user may resume execution from a different location.

Disconnection and migration prolong the execution time of transactions which results in a higher probability of conflicts with other transactions. Thus, it is necessary to ensure that transactions of mobile users are not penalized due to their extended execution time. This means that Long-Live Transactions (LLT) must be supported. Limiting MMDBMSs to purely ACID (Atomicity, Consistency, Isolation, Durability) may lead to too many aborts, which will result in a system that is perfectly consistent but gets only a small fraction of useful work done [12]. This dictates that MMDBMSs should support a range of correctness criteria. In addition, any solution should conform to multidatabase design restrictions, i.e., the autonomy of the local databases should not be violated.

There are two typical mobile computing architectures. In the General Mobile Computing Architecture, there is a fixed Mobile Support Station (MSS) that supports all mobile hosts (MHs) roaming within its cell. When an MH moves out of a cell and enters a new cell, it can no longer communicate with the previous cell's MSS and is under the control of the new cell's MSS. All MSSs communicate with each other via a fixed network. In the second architecture called Mobile Ad-hoc Network (MANET) Architecture, all MHs are roaming and the network that interconnects these MHs is a wireless network with a frequently changing topology, and there are no fixed infrastructure and no fixed MSSs. All MHs communicate with each other through a wireless medium. This second kind of architecture is widely used in battlefields and disaster recovery situations [15, 25] where it is not feasible to have a fixed network infrastructure.

Much research in the area of mobile database transaction management was based on the first architecture [11, 12, 26, 27, 30], while none on the second one. Supporting database transaction services in an ad-hoc mobile network raises new issues. If an MH stores a database, then other MHs will try to submit transactions and get data from it. In this environment both the user and the data source will be moving. So finding a route from one MH to another MH is necessary before submitting a transaction. Moreover many applications in this environment are time-critical which require their transactions to be executed not only correctly but also within their deadlines. Thus the Transaction Manager at the MH where the database is stored has to consider the mobility of the submitting MHs as well as the deadlines of the transactions. Another important issue in mobile ad-hoc networks is power or energy restriction on MHs because MHs are not connected to direct power supplies, and many of them will run on small and low-power devices. So energy-efficient solutions are needed for this environment. Such solutions should aim at providing a balance of energy consumption among MHs so that MHs with low energy do not run out of energy quickly, and thus the number of MH disconnections due to energy exhaustion can be reduced.

The goals of this paper are to propose an overall solution for managing database transactions in MANET that takes the issues of MH disconnection, migration, energy limitation, and transaction timing constraints into consideration, and perform simulation experiments to study the performance of the proposed solution. The paper presents algorithms that provide answers to the following questions: how to distribute a transaction initiated by a client, how to choose a server to process a transaction, how to schedule a transaction for execution, how to guarantee the ACID properties for a transaction, and how to return the transaction results to the client. The rest of the paper is organized as follows. Section 2 reviews some of the most recent mobile transaction management techniques. Section 3 describes the proposed MANET architecture and applications. Section 4 presents our transaction management technique that addresses the above issues for the proposed architecture. Specifically, Section 4 describes the information to be stored at each MH, the properties and classification of transactions, three different modes of energy to be used, how transactions are processed by MHs, and how transaction concurrency control and commitment are handled. Sections 5 and 6 present the simulation model and results. Finally Section 7 concludes the paper.

2 Related work

A number of transaction management techniques have been proposed for the General Mobile Computing Architecture. The technique presented in [30] is based on agent-based distributed computing. An agent is an object that encapsulates data and procedures that the receiving computer executes. A global transaction can be visualized as an agent that consists of sub-agents. Agents may be submitted from various sites including mobile stations. Agent-based computation is decentralized as the agents themselves communicate with each other in order to provide consistent and reliable computing. A set of structural dependencies allows the user to define compensating methods that are executed to compensate for already committed methods. In order to support migration, relocation points are pre-defined within the agent. The executions of agents can be isolated from each other by ensuring that concurrent execution occurs within the pre-defined breakpoints.

The Kangaroo model presented in [12] is based on the Open Nested model and captures the movement behavior of MHs. A global transaction (referred to as a Kangaroo transaction) consists of a set of Joey transactions, each consisting of all operations executed within the boundaries of one MSS. Each Joey transaction consists of one or more sub-transactions. Joey transactions may be committed independently. Kangaroo transactions execute in two different modes: under the Compensating mode, the failure of any Joey transaction causes all committed Joeys to be compensated and any other active Joeys to be aborted. Under the Split mode, all committed Joeys will not be compensated and the decision to commit or abort any active Joeys is left up to the component Data Base Management Systems (DBMSs).

The Pre-Commit model proposed in [26] introduces a pre-read, pre-write, and pre-commit operation to address the issues of mobile computing. Transactions of mobile users include read or pre-read data values, manipulate the data that have been read and pre-write the modified values stored at the MH. Once all pre-write values have been declared, the transaction pre-commits. At this point, all pre-write values are transmitted to the MSS, which then completes the transactions. A pre-write does not update the state of the physical data object but only declares its modified value. Once a transaction pre-commits, its pre-write values are written to a pre-write buffer maintained in the MSS and are made visible to other concurrent transactions executing at that MH and the respective MSS.

In the PSTM technique proposed in [9, 11], the Global Transaction Manager (GTM) consists of two layers: the Global Transaction Coordinator (GTC) which resides at each MSS, and the Site Transaction Manager (STM) which resides at each local database site. All local databases are connected to a fixed network. When an MH submits a global transaction to the GTC, the GTC creates a global data structure to supervise the overall execution of the global transaction. Then it submits all sub-transactions of the global transaction and their compensating transactions to the corresponding sites. The STM at each site supervises the execution of site transactions submitted at that site. After the completion of each site transaction, the STM informs the GTC about the status of the site transaction. If an MH migrates to a new cell, the MH will inform the new MSS about the identity of the previous MSS. The GTC of the new MSS will obtain the whole global data structure from the previous MSS and will be responsible for the execution of the global transaction of the migrated MH. If a user disconnects, its status is marked as disconnected but its transactions are not halted. But if the GTM determines that a catastrophic failure has occurred, then they are halted and marked as suspended.

The Multi-version transaction model proposed in [27] uses versions to increase data availability in a mobile environment. Each transaction is this model is either in start, committed or terminated state. A transaction can start and commit at MH but it will terminate only at MSS. The proposed scheme improves concurrency of mobile transactions by making use of the time between the commitment of transactions at MH and the termination of transactions at MSS. The scheme synchronizes the read/write lock requests on different versions of data using timestamps.

All the above reviewed techniques are based on the General Mobile Computing Architecture; so they address the mobility of users only. But in MANET the servers that store the data sources are also MHs. There are no fixed MSSs in this architecture, and the precise positions of the users and the data sources also cannot be located in advance. So before submitting a transaction, an MH has to find the MH which has the data. After processing the transactions the MH has to find the requester and submit the results. So location management should also be addressed while managing transactions in this environment. From the energy point of view, since all MHs will be running on limited power, they can go into doze mode or sleep mode at any time to reserve energy. The reviewed techniques do not address the energy-related issues, for example, if an MH goes into doze mode then how the MSS will take care of the transactions submitted by that MH. In our environment the servers called Large Mobile Hosts (LMHs), which store the entire DBMS and are capable of processing transactions as defined in Section 3, can also have energy limitation and are different from the MSSs which are servers with unlimited power. The reviewed techniques also do not deal with real-time transactions. Associating deadlines with transactions will have an impact on each of the proposed techniques. We will need a transaction scheduler and a commit protocol which take transaction types (firm and soft) and deadlines into consideration in order to minimize the number of transactions that must be aborted due to deadline violations. An efficient sub-transaction deadline assignment is also needed to carefully distribute global transactions' deadlines among their sub-transactions. In summary, none of the reviewed techniques can be applied directly to our environment. In the following sections we will describe our proposed architecture and applications and present a transaction management solution that fills in the gap in the reviewed techniques.

3 Proposed architecture and applications

In MANET, MHs communicate with each other without the help of a static wired infrastructure. So we have defined our architecture for this environment as illustrated in Fig. 1. Our architecture for MANET is adopted from the group mobility model defined in [15]. Depending on communication capacity, computing power, disk storage, memory size and energy limitation, MHs in this architecture can be classified into two groups: (1) computers with reduced memory, storage, power and computing capabilities (e.g. wearable computers developed at CMU [34]), which we will call *Small Mobile Hosts* (SMHs), and (2) classical workstations equipped with more storage, power, communication and computing facilities than SMHs, which we will call *Large Mobile Host* (LMHs). Every MH has a radius of influence. An MH can directly communicate with other MHs which are within its radius of influence. In Fig. 1, an oval shape with borders in dotted line represents the radius of influence of an MH. The communication link between two MHs is shown with dark dotted lines. Two MHs that are outside each other's radius of influence will be able to indirectly communicate with each other in multiple hops using other intermediate MHs between them [3]. For example, in Fig. 1, SMH 7 will not be able to communicate directly with LMH 1 because LMH 1 resides outside of the radius of influence of SMH 7, but it can indirectly communicate in multiple hops using SMH 6 and SMH 5 between them. Due to energy and storage limitations, we will assume that only LMHs will store the whole 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.

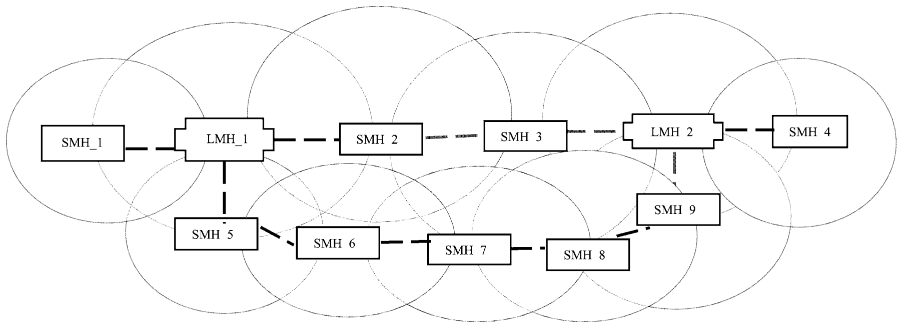

Fig. 1 Architecture

The proposed architecture can be used to support many applications, such as battlefields and disaster recovery. In battlefields, soldiers equipped with portable computing and transceiver devices can be considered as SMHs while battery-supported computers transported by humvees and tanks can be considered as LMHs. Both humvee and tank computers are more static than SMHs [25] and can store tactical information regarding enemy and other units in their databases. Soldiers can communicate with tank and humvee computers via wireless LAN technology to get the information from their databases (for example, which unit of enemy is located where, what is its strength).

In a disaster recovery situation such as an earthquake rescue operation, rescuers can be viewed as SMHs and mobile hospitals as LMHs. A mobile hospital can store the information of medical equipment in its database and rescuers can query about the inventory and inform the mobile hospital to keep certain arrangements ready for a particular patient whom they have found at their sites. There can be multiple mobile hospitals and the mobile hospitals can exchange information among themselves.

Other possible uses of MANET include Mobile ad-hoc voting [36] developed at MIT to spontaneously vote on issues across a mobile network and immediately check the results to see the impact of election, students using laptop computers to participate in an interactive lecture [37], business associates sharing and accessing databases in a meeting where it is difficult to setup a wired network.

4 Proposed transaction management solution

In this section, we present our solution to manage real-time database transactions in the architecture proposed in Section 3. We first describe some key information stored at the mobile hosts that are needed in order to understand the details of our solution. We then present the properties and classification of transactions to be processed by our solution and its employment of three modes of energy. We then describe how transactions flow between MHs, how an SMH processes transactions initiated at its site, how an LMH executes transactions sent to it by SMHs and other LMHs, how an LMH schedules transactions in its queue for execution, how it commits a compensatable/non-compensatable transaction, and how it verifies whether a transaction conflicts with other active transactions when the transaction wants to commit in order to decide whether the transaction should be committed/aborted.

4.1 Key information stored at mobile hosts

In our presented architecture, each MH will store some key information in its local database. The *ID* field will uniquely identify an MH. Every MH will get its coordinates from GPS (Global Positioning System) [20] periodically and store them in the *Position* field, which will be used at the time of routing a transaction from a source MH to a destination MH. Each MH will also store its *Radius of transmission range* in its local database. The *Energy_availability* field will record the amount of energy available in that MH at that point of time. This information is needed to identify the MH with the highest available energy at any point in time. Each LMH will maintain a *Global Schema*, which is the integration of all local schemas from all

LMHs and is used to identify which data object is stored in which LMH. Each LMH will periodically broadcast its *ID*, *Position* and *Energy_availability,* and the SMHs and other LMHs will record these in an *LMH_list* in their local databases after listening to the broadcast channel. The frequency of broadcasts depends on the application. If the application requires LMHs to move more often, then their states should be broadcast frequently.

4.2 Transaction properties and classification

In our real-time environment, transactions have deadlines and are classified into two categories: firm and soft [32]. When a firm transaction misses its deadline, its value becomes zero and it must be aborted (this definition is the same as that of a true hard real-time transaction as given in [38]). For example in the battlefield, a transaction that is trying to recognize a moving object of enemy (e.g. an enemy airplane) by comparing it with the images (signatures) stored in the database can be treated as a firm transaction. From the value function describing tasks with soft deadlines in [1], we can define soft transactions with two deadlines. A soft transaction still can be executed after its first deadline expires, but its value decreases after the first deadline and becomes zero after the second deadline. In a battlefield environment, querying the position of an enemy stored in the database can be treated as a soft transaction with two deadlines. Since the position of the enemy will be changing frequently in this environment, the value of this transaction will decrease over time and will become zero after a certain time. In this case, the first deadline will be the point of time when the probability of getting the enemy at that position starts decreasing due to mobility of enemy. And the second deadline will be the point of time when the probability of getting the enemy at that position is zero.

A global transaction is defined as a transaction that requires data items from different sites. The part of a global transaction, which is executed at a particular site, is defined as a sub-transaction. We have assumed that for each global transaction there will be no more than one sub-transaction for one site. Depending on whether transactions can be compensated or not after they have been committed, they are categorized as compensatable and non-compensatable transactions. In the battlefield, updating the location of enemy is an example of a non-compensatable transaction because if the location of enemy is updated with wrong information, then a soldier can kill another soldier of his own group who happens to be in that location by mistake, and this cannot be compensated. A transaction that marks the status of a soldier (alive/dead) in the database can be considered compensatable.

Considering transaction criticality with respect to transaction commitment, sub-transactions of a global transaction can be classified as vital and non-vital sub-transactions. All vital sub-transactions of a global transaction must succeed for the global transaction to succeed. An abort of a non-vital sub-transaction does not require a global transaction to be aborted. For example, in the battlefield, a soldier can ask for images of different objects (enemy troops, rivers, roads, terrains, etc.) from the databases stored in tanks and humvees. In this type of transactions, the soldier may specify that the image of one object (e.g. enemy troop) is vital and the image of another object (e.g. road) is non-vital. This means that the image of enemy troop is more important to the soldier.

4.3 Handling energy limitation

Our transaction management solution is aimed at reducing energy consumption at each MH and providing a balance of energy consumption among MHs so that MHs with low energy do not run out of energy quickly, and thus decreasing the number of MH disconnections due to energy exhaustion. To reduce energy consumption, each MH will operate in three modes, Active, Doze, and Sleep, as follows:

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 at 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 [4].
3. Sleep Mode: Both the CPU and the communication device of the MH are suspended. This is equivalent to a system failure considered in classical transaction processing work.

We have considered all the three modes of mobile hosts in our proposed solutions for managing transactions at SMHs and LMHs as described in Sections 4.5 and 4.6.

4.4 Overall transaction flow between MHs

We assume that when an SMH initiates a transaction, it will send the entire transaction to an LMH to process before it moves away. This LMH will act as a coordinator for this global transaction. Then the coordinator LMH will check the global schema to find which data item is stored at which LMH and divide the global transaction into sub-transactions (or also called site-transactions) in such a way that all data required by a sub-transaction resides at only one LMH, which is called the participant LMH of the global transaction. The coordinator LMH will submit the sub-transactions to the respective participant LMHs for execution. Then the coordinator LMH with the help of the participant LMHs will commit/abort the global transaction and return the result to the requesting SMH. In the next two Sections 4.5 and 4.6, we describe in detail how an SMH processes the transactions initiated at its site and how an LMH executes the transactions once it received them from an SMH or from another LMH.

4.5 Handling transaction submission from SMH to LMH

Since in our architecture MHs have energy limitation and transactions have timing constraints, we need to have a transaction management policy that provides a balance of energy consumption among MHs to reduce the chance that MHs get disconnected due to power exhaustion, while at the same time reduces the number of transactions that must be aborted due to missing their deadlines. Here we consider time as the most important factor in handling firm transactions and energy in handling soft transactions. So the SMH will submit its firm transactions to the nearest LMH so that the transactions can meet their deadlines. But for soft transactions, the SMH will submit them to the LMH which has the highest available energy. Here we are sacrificing the first deadlines of soft transactions in favor of balancing the energy consumption because soft transactions can still be executed after their first deadlines have expired. Thus when an SMH initiates a firm transaction, it will search its local database to identify the nearest LMH, find a route to this nearest LMH using some route discovery scheme and submit the transaction to this LMH for processing. We can use the LAR Scheme 2 proposed in [20] to find a route because this technique needs only the position of the destination, which is available in the local database of each SMH.

Now if the nearest LMH is in active mode, it will process the transaction. If it is in doze mode and if the transaction is firm, it will wake up and process the transaction in order to reduce the chance that the transaction will miss its deadline. But if the LMH is in sleep mode, the requesting SMH will wait for some time to get the result of the submitted transaction. If it does not receive the result of the transaction in this time period, it will assume that the nearest LMH is either in sleep mode or disconnected. So it will again check its local database to find the next nearest LMH, find a route to this LMH and submit the transaction. The SMH can determine the amount of time it must wait for the LMH to return the transaction results using the runtime estimate of the transaction, communication overhead and possible delay due to disconnection obtained from the transaction history as follows.

Let *Runtime estimate of the transaction* $=$ $\boldsymbol{r}$ *time units*
Estimated communication time for submitting the transaction to the LMH and getting back the result $=$ $\boldsymbol{c}_1$ *time units*
Average estimated communication cost for a sub-transaction $=$ $\boldsymbol{c}_2$ *time units*
Number of sub-transactions $=$ $\boldsymbol{n}$ (*from history*)
Total Communication cost $= (\boldsymbol{c}_1 + \boldsymbol{n}\, c_2)$ *time units*
Then ***Waiting_period*** $= \boldsymbol{r} + (\boldsymbol{c}_1 + \boldsymbol{n}\, c_2)$

If the transaction is soft, the SMH will search its local database for the LMH with the highest available energy, find a route to this LMH and submit the transaction to it. If the SMH does not get the result of the transaction after some time, it will search the local database to find the LMH with the next highest available energy. If the requesting SMH moves after submitting a transaction to an LMH, it will inform the LMH of its new position. The algorithm is shown in Fig. 2.

4.6 Handling transaction processing at LMH and result submission to SMH

An LMH can receive global transactions from an SMH or sub-transactions from other LMHs. Each LMH has three components: (1) Transaction Scheduler (TS) which schedules all global transactions and sub-transactions stored in the LMH's transaction queue; (2) Transaction Coordinator (TC) which divides the global transaction into sub-transactions and submits them to corresponding LMHs, and returns the results to the requesting MH; and (3) Transaction Manager (TM) which manages the execution of sub-transactions. After receiving a transaction from an SMH, if the LMH is in active mode, it will pass the transaction to the TS. If the LMH is in doze mode and if the transaction is firm, it will wake up and pass the transaction to TS. But if the transaction is soft, the LMH will not wake up. Here we are sacrificing soft transactions for energy consideration because the LMH will usually go into doze mode to reserve

```
Suppose an SMH whose ID is SMH_ID and position is SMH_Position initiates a Transaction T1 with type T1_type,
deadline T1_deadline

Submit_transaction_from_SMH_to_LMH (T1, T1_type, T1_deadline, SMH_Position, SMH_ID)
Begin
        If T1_type is firm
                Search the LMH_List  to get the nearest LMH that is not yet visited (LMH_c)
        Else
                Search the LMH_list to get the LMH with highest energy and not yet visited (LMH_c)
         End If
        Find a route to LMH_c
        Submit T1 to LMH_c
        Set Waiting_period = value
        Set timer =Waiting_period
        While timer ≠ 0 do
             If the result of T1 is received
                Find a route to LMH_c
                Send an acknowledgement to LMH_c
                Exit
           End If
          timer--
        End While
        T1_deadline = T1_deadline – Waiting_period
        If T1_deadline = 0          // T1 has missed the deadline
           Abort T1
        Else
           Mark LMH_c as visited
          Submit_transaction_from_SMH_to_LMH (T1, T1_type, T1_deadline, SMH_Position, SMH_ID)
        End If
End
```

Fig. 2 SMH execution algorithm

its energy, and for soft transactions, they still will be executed after they missed their first deadlines.

The TS at LMH will use a real-time energy-efficient dynamic scheduling algorithm (presented in the next Section 4.7) to schedule transactions that it received. When executing a transaction, the TC will pass the transaction to the LMH's TM if the transaction is a local transaction. Otherwise it will divide the global transaction into sub-transactions, initiate these sub-transactions on the remote participant LMHs, and distribute the deadline of the global transaction among the sub-transactions using a deadline distribution algorithm such as the EQF Strategy proposed in [17]. Then the TC will use the Commit Protocol and Concurrency Control Protocol (presented in Sections 4.8 and 4.9) to decide the fate of the global transaction. After the TC has decided to commit a transaction, it will submit the result to the requesting SMH. If the requesting SMH is in active mode, it will receive the result. If it is in doze mode and if the transaction is firm, then in order to meet the deadline of the transaction, the SMH will wake up and receive the result. But if the transaction is soft, it is up to the SMH to decide whether it should come into active mode and receive the result or remain in doze mode to reserve its energy for firm transactions.

If the requesting SMH is in sleep mode, it will not be able to receive the result. So if the TC does not receive any acknowledgement till the deadline of the transaction, it will abort the transaction if the transaction is firm. Otherwise, the TC will calculate the slack time using the second deadline of the transaction. If this slack time is zero, it will abort the transaction. Otherwise it will divide the slack time into some time intervals and will submit the result again to the requesting SMH during those intervals. The motivation behind this technique is that since the transaction is soft and transmission consumes a significant amount of energy, the LMH will not continuously keep sending the result to the sleeping SMH and lose its energy. The length of the time-interval is calculated as follows.

Let the second deadline of the soft transaction $= \boldsymbol{d}_2$ *time units and Current time* $= \boldsymbol{t}$ *time units*
Slack time $\boldsymbol{s} = (\boldsymbol{d}_2 - \boldsymbol{t})$ *time units*
Average Energy level of LMH $= \boldsymbol{E}$ *energy units*
Average energy consumption to process a sub-transaction $= \boldsymbol{m}$ *energy units*
Average energy consumption to process a global transaction $= \boldsymbol{M}$ *energy units*
Average energy consumption to transmit a message $= \boldsymbol{k}$ *energy units*
The average number of sub-transactions on an LMH $= \boldsymbol{n}$
The average number of global transactions on an LMH $= \boldsymbol{N}$
Total Energy Consumption for all transactions on an LMH $= (\boldsymbol{m} * \boldsymbol{n} + \boldsymbol{M} * \boldsymbol{N})$
Average Remaining Energy $\boldsymbol{R} = \boldsymbol{E} - (\boldsymbol{m} * \boldsymbol{n} + \boldsymbol{M} * \boldsymbol{N})$
Thus the total number of intervals $\alpha = \boldsymbol{R}/\boldsymbol{k}$
And the length of an interval $= \boldsymbol{s}/\alpha$

The *LMH* execution algorithm is captured in Fig. 3(a) and (b).

```
Suppose an LMH receives a transaction T with ID T_ID, transaction type T_type, deadline T_d, Runtime estimate
T_e, Requester ID R_ID, Requester position R_pos, Requester energy R_energy, Data item List L.
// This module will be executed when an LMH receives a transaction
Execute_transaction_at_LMH (T, T_ID, T_type, T_d, T_e, R_ID, R_Pos, R_energy, L)
Begin
     Schedule (T_ID, T_type, T_d, T_e, R_energy) and take the first transaction Tf from the queue
    If Tf_ID is for a sub-transaction     // Tf is a local transaction
        Execute the sub-transaction Tf.
        Submit_result_to_the_requester(Tf, R_ID, R_Pos)
    Else    // Tf is a global transaction
        Execute global transaction (Tf, Tf_ID, Tf_type, Tf_d, Tf_e, R_ID, R_Pos, R_energy, L)
        While LMH has not received Ack from requesting SMH and slack time for 1st deadline of Tf > 0 do
            If LMH has received an Ack
                Commit Tf
                Remove Tf from the active transaction list
                Exit
            End If
        End While
        If LMH has not received an Ack from requesting SMH  // SMH is in sleep mode
            If Tf_type is firm
                Abort the transaction.
                Submit_result_to_the_requester (Tf, R_ID, R_Pos)
            Else    // Transaction type is soft
                Find the slack time using the second deadline
                If slack time =0
                    Abort the transaction
                    Submit_result_to_the_requester (Tf, R_ID, R_Pos)
                Else
                    Divide_slack_time_and_submit(Tf, Tf_d,  slack_time, R_ID, R_Pos )
                End If
            End If
        End If
    End If
End
```

Fig. 3 (a) LMH execution algorithm (part 1)

```
// This module will be executed when the transaction is a global transaction
Execute_global_transaction (T, T_ID, T_type, T_d, T_e, R_ID, R_Pos,  R_energy, L)
Begin
        Search the global schema.
        Get the LMH_list for Data item List L.
        Divide the Global Transaction into sub-transactions.
        (Refer to this list of sub-transactions as S_list)
        Distribute deadline T_d among  S_list using EQF Strategy [Kao,1993]
        For each LMH in the LMH_list
                Find a route to LMH using LAR Scheme 2 [Ko, 1998]
                Submit the corresponding sub-transaction from S_list to this LMH
        End For
        Wait until all the vital sub-transactions are completed
        Run PGSG Algorithm [Dirckze, 1998,2000] to check Atomicity and Isolation (A/I)  property violations
                                                    //Details of PGSG are given in Section 4.9
        If PGSG Algorithm's outcome is to abort T due to A/I violations
                If slack time of T  is greater than 0  // For soft, slack time is calculated using second deadline
                        Execute_transaction_at_LMH (T, T_ID, T_type, T_d, T_e, R_ID, R_Pos, R_energy, L)
                Else
                        Abort T
                End If
        Else
                Submit_result_to_the_requester(T, R_ID, R_Pos)  // PGSG 's outcome is to commit T
        End If
End

// This module submits the result of the transaction to the requesting SMH
Submit_result_to_the_requester (T, R_ID, R_Pos)
Begin
        Find a route to the requester
        Send result of T to the requester
End

//This module divides the slack time into intervals and submit result to the requester in these intervals
Divide_slack_time_and_submit(Tf, Tf_d, slack_time, R_ID, R_Pos )
Begin
        Divide the slack time into $\alpha$  intervals
        Loop_count = 0
        While Loop_count is less than $\alpha$ and LMH has not received an Ack from  requesting SMH do
                Submit_result_to_the_requester (Tf, R_ID, R_Pos)
                Set WaitingTimePeriod = interval   // max .time that LMH waits for an Ack from requesting SMH
                While LMH has not received an Ack and WaitingTimePeriod$\neq$0 do
                        If LMH has received an Ack from requesting SMH
                                Commit Tf
                                Remove Tf from active transaction list
                                Exit
                        End If
                End While
                Increment Loop_count
        End While
End
```

Fig. 3 (b) LMH execution algorithm (part 2)

4.7 Energy-efficient real-time transaction scheduling

Each LMH will schedule transactions it receives from SMHs and other LMHs for execution. The scheduling algorithm has to consider not only transaction types (firm and soft), transaction deadlines, but also the energy limitation of the MHs. Here we can use the Least Slack (LS) cognizant technique proposed in [2] with certain modifications with respect to energy constraints, disconnections and transaction types. In the LS technique, transactions with less slack time are scheduled before transactions with more slack time. Our scheduling algorithm follows the same principle but calculates the slack time s of a transaction based on its deadline d, runtime estimate c, probability of disconnection during execution Pd, and average time loss due to disconnection Td using the following formula:

$$s = d - (t + c + Pd * Td) \tag{1}$$

The values of Pd and Td can be obtained from the transaction execution history as follows. The LMH can keep track of how many times each SMH had been disconnected when the LMH wanted to submit the result of a transaction to it and what was the duration of the disconnection. From these values, the LMH can determine the probability of disconnection and the average time loss due to disconnection for a particular SMH.

If two firm transactions have the same slack time, then a higher priority will be given to the one whose requester has less energy available in order to reduce the chance that the requester will run out of energy before the transaction can be completed. The MHs while submitting their transactions/sub-transactions to the LMH will also send their energy levels to the LMH. The same policy will be adopted if two soft transactions have the same slack time. If the slack time of a firm transaction is equal to the slack time of a soft transaction, then a higher priority will be given to the firm transaction considering that the firm transaction will be aborted if it misses its deadline. Now if the slack time of a soft transaction is found to be negative, then its slack time will be recalculated using its second deadline and its priority will be recalculated. If the recalculated slack time is again found to be negative, then the transaction will be discarded. The algorithm is captured in Fig. 4.

4.8 Transaction commitment

For committing a global transaction, we need a Commit Protocol that considers the host mobility, energy limitation and the time constraint characteristics of our environment. So our primary aim is to reduce the number of communication messages between the coordinator LMH and the participant LMHs when committing a global transaction. The motivation behind this idea is threefold. Firstly fewer communication messages means less transmission from a mobile host, which means less energy consumption because transmission consumes a substantial amount of energy. Secondly if we can commit a transaction with fewer messages, then there is a higher chance that we will meet the deadline of the transaction. Finally since the hosts will be mobile in our environment, the commit protocol with less communication will have fewer disconnections.

We propose to use Semantic Atomicity [23] to handle commitment of compensatable transactions, that is, we allow them to commit if all their vital sub-transactions are committed. This will allow the transactions to commit early. As a result, resources will be released early and LLTs will be supported. In order to take care of the non-compensatable transactions, we can relax the local autonomy requirement by adding the "Pre-commit" stage in our commit protocol. If a participant LMH wants to commit a non-compensatable sub-transaction, first it should pre-commit at its site.

```
Schedule (T_ID, T_type, T_d, T_e, R_energy)
Begin
        Calculate the slack time for all transactions using Formula 1.
        If the slack time of a transaction T is less than 0 and T_type is firm
                Remove T from the queue and discard it.
        End if
        Sort all the transactions that have slack time > 0 according to their slack times.
        Assign higher priorities to transactions with shorter slack times.
        If two firm transactions or two soft transactions have the same slack time
                Give a higher priority to the one whose requesting MHs has less energy.
        End if
        If the slack time of a firm transaction is equal to the slack time of a soft transaction
                Give a higher priority to the firm transaction.
        End if
        If the slack time of a soft transaction is negative
                Recalculate the slack time using its second deadline and Formula 1.
                If the recalculated slack time of a soft transaction is negative
                        Discard the soft transaction.
                Else
                        Recalculate its priority.
                End if
        End if

End
```

Fig. 4 Energy-efficient real-time transaction scheduling algorithm

It can commit the sub-transaction only after getting the "Commit" message from the Coordinator LMH.

Case 1: Transaction is compensatable

The participant LMHs can commit the sub-transactions locally at their sites. After committing the sub-transactions, they will inform the coordinator LMH about their decision. When the coordinator LMH gets all the vital sub-transactions committed and if the PGSG algorithm verified that the global transaction does not conflict with any active transaction, it can commit the global transaction. If any of the vital sub-transactions is aborted, the coordinator LMH will abort the global transaction and ask the participants which have committed their sub-transactions to run compensating transactions for those sub-transactions.

Case 2: Transaction is non-compensatable

The participant LMHs will pre-commit the sub-transactions at their sites and inform the LMH Coordinator about their decision. If the coordinator LMH finds all the vital sub-transactions pre-committed and if the PGSG algorithm verified that the global transaction does not conflict with any active transaction, then it will commit the global transaction and inform all the participants about its decision. If the coordinator LMH finds any of the vital sub-transactions aborted, it will send an Abort message to all the participants. If the participant LMH gets a Commit message from the coordinator, it will commit the sub-transaction; if it gets an Abort message, it will abort the sub-transaction. When any data item is in the pre-committed state, no other transactions can use the data item because it may lead to cascading aborts.

4.9 Transaction concurrency control

To verify the Isolation property of compensatable transactions we adopt the Partial Global Serialization Graph (PGSG) concurrency control algorithm that has been developed for the General Mobile Computing Architecture by [9, 11]. This algorithm uses the optimistic concurrency control principles, which means data conflicts (Isolation property violations) caused by a transaction are not checked during its execution but during its commit time. However, the algorithm provides a way to minimize unfair treatment of long-lived mobile transactions by allowing their vital sub-transactions to commit independently without knowing the fate of their global transactions, and by allowing a global transaction to be verified for the Isolation property violation once its vital sub-transactions have committed. So first the requesting SMH has to define the vital and non-vital [7] sub-transactions when it submits a global transaction to an LMH. When all the vital sub-transactions are completed, the coordinator LMH can run the PGSG algorithm to verify the Isolation property based on the concept of serialization graphs. Each participant LMH will maintain a site serialization graph, which shows the order of execution of all sub-transactions at that site. When the coordinator LMH wants to verify the Isolation property for a global transaction, it will ask for a Predecessor Graph for the global transaction from all the participating LMHs. Then the coordinator LMH will construct the PGSG by merging all the Predecessor Graphs, and check for cycles in the PGSG. If there is no cycle, then the Isolation property is not violated; otherwise the Isolation property is violated. Then it will propagate the PGSG graph to the participating LMHs in order to guarantee serializability [11].

If the Isolation property is not violated, then the transaction is *toggled* and its execution continues until all its remaining non-vital sub-transactions are either committed or aborted. The coordinator LMH can then commit the transaction and submit the result to the requesting SMH. A *toggled* transaction is guaranteed not to be aborted due to concurrency conflicts because all its vital sub-transactions have been committed unless it obstructs the execution of another global transaction while in a Suspended state (note that PGSG assumes that all transactions and sub-transactions are compensatable; this is one of the limitations of this algorithm that we intend to examine further as a part of our future research discussed in Section 7). A transaction is said to be in a Suspended state (i.e. the transaction is halted, no new sub-transaction can be initiated) if its MH is disconnected from the network. Normally, if such a failure occurs, the transaction should be aborted. However, since in our environment, there are many MHs that may be disconnected from the network, this determination may be incorrect. In order to minimize erroneous aborts, Suspended global transactions are not aborted until they obstruct the execution of other global transactions. When the MH reconnects to the network, its transaction comes out of the Suspended state. The design of *toggled* transactions addresses the issues of disconnection and migration of MHs that cause prolonged execution of mobile transactions. The design of Suspended transactions addresses the issue of disconnection due to catastrophic failures.

If the Isolation property is violated and the transaction has not yet missed its deadline (or its second deadline if it is a soft transaction), the coordinator LMH will abort the transaction, and then restart the transaction. But if the Isolation property is violated and the transaction has missed its deadline (or its second deadline if it is a soft transaction), the coordinator LMH will abort the transaction and send the result to the requesting SMH. An abort of a global transaction requires all participant LMHs to run the compensating sub-transactions for its sub-transactions that have been locally committed. PGSG has been proved to correctly verify the Isolation property of all transactions and incur little overhead [11].

5 Simulation model

We study the performance of our proposed solution by means of simulation. The solutions we have discussed in Section 4 depend on many factors (e.g. the ratio of firm and soft transactions, deadlines of transactions, number of SMHs and LMHs, power level of SMHs and LMHs, routing protocols, speed of SMHs and LMHs, data conflicts during execution). It is difficult to correctly formulate their relationship using an analytical model without making many unrealistic assumptions. That is why we have decided to analyze the performance of our solution using simulation—an approach that many existing works have adopted [6, 11, 14, 18, 21, 22, 24]. As we did not find any standard simulation model or benchmark in the literature specially built for database transaction processing in MANET, we have built our simulation model using simulation parameters taken from different published articles as presented in Section 5.2. The simulation model is implemented using the AweSim simulation tool [31]. The initial energy levels of all LMHs are identical, and the same holds true for SMHs. The locations of MHs are assumed to be inside a 1000×1000 square units region [20]; their initial locations (x and y coordinates) are obtained using a random distribution within the region. All MHs are in active mode initially. When an MH is not generating any transaction or not processing any transaction, it goes into doze mode. When the power of any MH goes down to zero, its mode is changed to sleep mode. The energy consumption for transition of modes of MHs is not considered. We also assume that every MH can communicate with other MHs, either directly or via multiple hops. To validate our simulation model, we have conducted experiments and analyzed whether we are getting the expected results or whether we can justify the obtained results. For example, if there are more LMHs in the system, then fewer transactions missing deadlines are expected. So we have varied the number of LMHs in the system and checked the number of transactions missing deadlines. Similarly, we have varied the rest of the simulation dynamic parameters in a wide range of possible values to observe the behaviors of our solution under different operating conditions. Note that the purpose of our simulation experiments is to study the relative performance of different solutions when certain parameters are varied to answer the questions such as “Will a specific algorithm cause more transactions to miss their deadlines or consume more energy or achieve more balance in energy consumption distribution among mobile servers when a parameter X increases?” This kind of answer will help the users decide whether they should increase a particular parameter X (e.g. number of servers) when using the algorithm. Our purpose is not to obtain an absolute performance.

5.1 Performance measurements

The performances of the proposed techniques are measured in terms of the percentage of transactions missing deadlines, the energy consumption by mobile hosts, and the average difference in energy consumption between two LMHs using the following equations:

$$\%\,\mathit{Transactions\ missing\ deadlines} = \frac{N_{\text{missDeadline}}}{N_{\text{totalTrans}}} \times 100\%$$

where $N_{\text{missDeadline}}$ is the number of transactions missing their deadlines, and $N_{\text{totalTrans}}$ is the total number of transactions in the system.

$$\mathit{Overall\ Energy\ consumption} = \sum_{i=1}^{n} L_i + \sum_{i=1}^{m} S_i,$$

where n is the total number of LMHs, m is the total number of SMHs, and L_i and S_i are the energy consumption of LMH$_i$ and SMH$_i$, respectively. L_i and S_i are computed using the following equation:

$$L_i \quad \text{or} \quad S_i = (T_{\text{active}} \times P_{\text{active}}) + (T_{\text{doze}} \times P_{\text{doze}})$$

where T_{active} is the total time a MH$_i$ spent in active mode, P_{active} is the power dissipation rate of a MH$_i$ in active mode, and T_{doze} and P_{doze} are defined similarly for doze mode.

The average difference in energy consumption between two LMHs is computed to study the distribution of energy consumption among LMHs. Since we assume all LMHs have the same initial energy amount, an ideal transaction management technique should yield a balance of energy consumption in the system in order to reduce the chance of LMH disconnection due to power exhaustion. The average difference in energy consumption between two LMHs is computed as follows:

$$\mathit{Average\ Difference\ in\ energy\ consumption} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{(\boldsymbol{n}-1)\times \boldsymbol{n}}$$

where n is the total number of LMHs and x_i is the total energy consumption of LMH$_i$.

5.2 Simulation parameters

Tables 1 and 2 show the static and dynamic simulation parameters, respectively. Static parameters retain the same values throughout all the experiments. Dynamic parameters are those for which we want to conduct testing cases to study their effects on the overall system performance. While we are studying the effect of one dynamic parameter, the other dynamic parameters take their default values as listed in Table 2. Below we provide the explanations of some of the parameters.

Global Transactions are created with an exponential distribution of inter arrival times. This distribution is chosen because of its popularity in modeling the arrival of

Table 1 Static parameters of the simulation model

Parameter	Value	Reference	Parameter	Value	Reference
Bandwidth of wireless medium	100 kbps	[20]	Time to end one transaction	0.0054 ms	[8]
CPU power of LMH	140 MIPS 1700 MIPS	[8] [5]	Probability of read operations	60%	[8]
CPU power of SMH	4 MIPS 100 MIPS	[16] [5]	Power dissipation rate for LMH in active	170 W/h	[28]
Radius of influence of SMH	100 units[a]	[20]	Power dissipation rate for LMH in doze	20 W/h	[28]
Radius of influence of LMH	200 units[a]	[20]	Power dissipation rate for SMH in active	7 W/h	[16]
Main memory access time per word	0.00018 ms	[8]	Power dissipation rate for SMH in doze	1 W/h	[16]
Number of bytes per memory word	8	[8]	No. of sub-transactions per global transactions	TRIAG(3,4,5)	[9]
Time to preprocess one transaction	0.0072 ms	[8]	No. of operations per sub-transactions	UNIF(5,10)	[9]
Time to preprocess one operation	0.000007 ms	[8]			

[a]The parameters are also varied dynamically in the range of (50, 100) and (100, 200) for SMH and LMH, respectively, for additional experiments in Section 7.

Table 2 Dynamic parameters of the simulation model

Parameter	Default value	Value range	Reference	Parameter	Default value	Value range	Reference
Global transaction inter-arrival time	EXPON (0.2)	EXPON(0.2) to EXPON(10)	[33] [14]	MH moving speed	50	0–100	[14]
Probability that a transaction is Firm	0.5	0.1–1	[14]	LMH location error rate	0%	0–100%	[14]
Number of SMHs	40	20–40	[14]	LMH energy error rate	0%	0–100%	[14]
Number of LMHs	20	5–20	[14]	Slack factor	15	5–25	[14]

transactions in database applications. All MHs are assumed to have the same speed. The LMH location and energy error rates are the percentages of error in the location and energy information of an LMH, which is broadcasted to SMHs and other LMHs. The slack factor is used to calculate the global transaction deadlines as follows:

$$\textit{Transaction deadline} = \text{Transaction creation time} + \text{Run time estimate} * \text{Slack factor}$$

The runtime estimate for each global transaction is computed by estimating the time to preprocess the transaction, process all reads/writes operations in the transaction (each read/write requires one main memory word access since we assume that our database is memory resident); terminate the transaction, and communicate. The communication time is calculated as the time taken to transfer a message from the position of the SMH to the average position of the LMHs. The second deadline of a soft transaction is twice of its first deadline.

The bandwidth of the wireless medium is adopted from [20]. The CPU power, memory access time and word size are chosen based on the DEC 3000 model machine [8] for LMHs. All SMHs are assumed to have the same radius of influence, and the same assumption is made for LMHs. The additional experiments in which we used more powerful machines and generated the radius of influence of MHs randomly were conducted as reported in Section 7. The time to preprocess a transaction includes the time to add the transaction to the active transaction list and log "begin transaction". The time to preprocess an operation denotes the time to fetch one instruction. The time to end a transaction includes the time to remove the transaction from the active transaction list and move it to the committed/aborted transaction list. The power dissipation rates of an SMH and LMH are taken from [16, 28], respectively. The number of subtransactions for each global transaction is determined using the triangular distribution, which is usually used when the exact form of distributions is unknown [19]. The number of operations per sub transaction is calculated from the uniform distribution UNIF (5, 10).

6 Simulation results

For each set of the simulation experiments reported in the following Sections, 6.1–6.6, the results are calculated as averages of 20 independent runs. In each run, at least 1000 transactions are completed in the system. 90% confidence intervals are obtained and the width of the confidence interval of each data point is within 5% of the point estimate.

6.1 Evaluation of LMH assignment alternatives

To assign LMHs to handle transactions initiated by SMHs, as presented in Section 4.5, we have proposed that firm transactions are always submitted to the nearest LMHs and soft transactions to the highest energy LMHs. This technique is called Transaction Type Based Server Assignment (TTBSA). Its objective is to reduce the number of transactions missing their deadlines as well as to balance the energy consumption among the LMHs in the system by taking transaction types into consideration. We examine two other alternatives for LMH assignment. One alternative considers only LMH location and the other considers only LMH energy. These techniques are called Location Based Server Assignment (LBSA) and Energy Based Server Assignment (EBSA), respectively. In LBSA, all transactions initiated by SMHs are always sent to their nearest LMHs. In EBSA, all transactions initiated by SMHs are always sent to the LMHs that have the highest energy available at that time. We then perform a set of experiments to compare these three alternatives when the ratios of firm and soft transactions are varied.

From Fig. 5 we observe that when all transactions are submitted to the nearest LMH, the percentage of transactions missing deadlines is always lower than the one that incurs when all transactions are submitted to the highest energy LMHs or when firm transactions are submitted to the nearest LMHs and soft transactions to the highest energy LMHs. The average percentage of transactions missing deadlines is the lowest (42.76%) for LBSA and the highest (78.26%) for EBSA. The performance of TTBSA

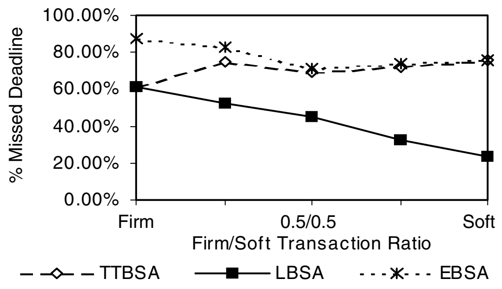

Fig. 5 Effects of firm/soft transaction ratio on % transactions missing deadlines

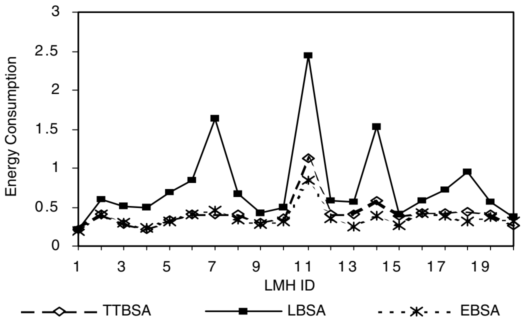

Fig. 6 Distribution of energy consumption in LMHs when ratio of firm/soft transactions is 50/50

is in between those of the LBSA and EBSA. On average the percentage of transactions missing their deadlines in TTBSA is 27% higher than that in LBSA and 8% lower than that in EBSA.

We have experimented with different ratios of firm and soft transactions in studying the energy consumption distribution among LMHs. Due to space limitation, we show the energy consumption in each individual LMH only for the case when the ratio is 50/50 (Fig. 6). In all experiments, we observe that EBSA gives the best-balanced energy consumption among LMHs, TTBSA gives the second best, and LBSA gives the worst.

We can conclude from the above results that when a system needs only to reduce the number of transactions missing deadlines, LBSA is the best technique. When a system only requires a good balance in energy consumption by LMHs, EBSA is the best technique. But when a system needs to reduce the percentage of transactions missing deadlines as well as to balance the energy consumption among the LMHs, then TTBSA should be the choice.

TTBSA is used in the rest of the simulation experiments reported in the Sections 6.2–6.6.

6.2 Varying number of LMHs

In this experiment, we have varied the number of LMHs for different mixtures of firm and soft transactions. Figure 7 shows that fewer transactions will miss their deadlines when there are more LMHs in the system. This is expected since LMHs are servers, and as there are more LMHs, transactions need less waiting time to use the servers, and thus fewer transactions will miss their deadlines. In Fig. 8, we observe that

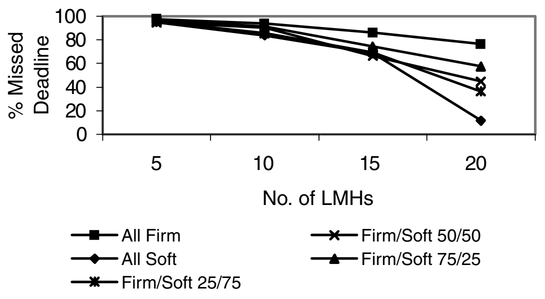

Fig. 7 Effects of number of LMHs on % transactions missing deadlines

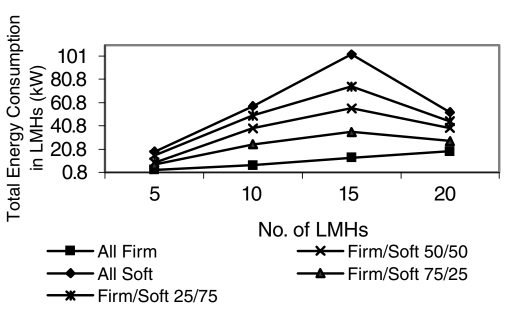

Fig. 8 Effects of number of LMHs on total energy consumption in LMHs

the total energy consumption in LMHs increases as the number of LMHs increases, except for the case when the number of LMHs is above 15 and there are some soft transactions in the system. This is due to the fact that when there are more servers, fewer transactions will be aborted, and thus LMHs will consume more energy to complete more transactions. The results also show that the more firm transactions the system has, the more transactions will miss their deadlines (thus the less *LMH* energy will be needed). This is because soft transactions are aborted only if they missed their second deadlines, while firm transactions are aborted as soon as they missed their deadlines. When the number of LMHs is high (above 15 in this experiment) and not all transactions are firm, the total energy consumption in LMH decreases. This is because when there are more LMHs to process transactions, each individual LMH will have fewer transactions in its queue, and thus it does not have to spend much time and energy on scheduling so many transactions, checking for conflicts between firm and soft transactions, and checking whether soft transactions have missed their first or second deadlines.

6.3 Varying inter arrival time of soft transactions

In this experiment, we have varied the inter-arrival time of soft transactions and calculated the energy consumption for each LMH. In our simulation model, the energy level of each LMH is updated when it has processed a transaction or a sub-transaction by calculating the time it was in active mode. But if another transaction enters the system before the energy level of the LMH is updated, then there is a possibility that the SMH, which initiated that transaction, will identify a wrong LMH as the one with the highest energy. That means if the inter-arrival time of transactions is small, i.e.

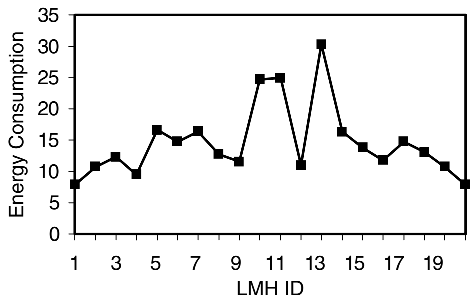

Fig. 9 Distribution of energy consumption in individual LMHs for soft transactions with inter-arrival time = EXPON(1)

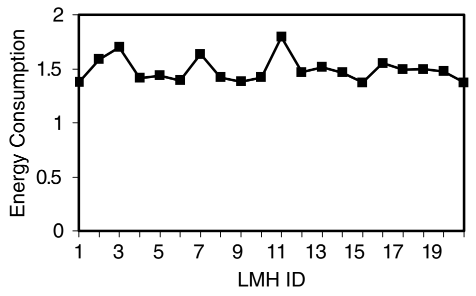

Fig. 10 Distribution of energy consumption in individual LMHs for soft transactions with inter-arrival time = EXPON(5)

the system load is high, then many SMHs will not have the recent energy levels of LMHs. As a result, they will not be able to correctly identify the LMH with the highest energy for soft transactions. So the total energy consumption in the system will not be evenly distributed among all the LMHs. But if the inter-arrival time is large, i.e. the system load is low, then the energy consumption in LMHs will be balanced. From Fig. 9, we can see that the energy consumption of LMHs is not uniformly distributed when the inter-arrival time is EXPON(1). But when the inter-arrival time increases to EXPON(5), the energy consumption of LMHs is almost uniform (Fig. 10).

6.4 Varying location error rate

In this experiment, we have varied the location error rate, which is the percentage of error in the location information of each LMH that is periodically broadcast to the SMHs and other LMHs. We have examined the case when only firm transactions exist in the system, as the SMHs will be using the location information of LMHs to manage firm transactions. Figure 11 shows that when the location error rate is not very high (below 80%), it does not have a severe impact on the system performance. By averaging all values shown in Fig. 11, we have found that on average, only 6% more transactions missed their deadlines when the LMH location information was less than 50% accurate. However, when the error rate was extremely high (above 80%), more transactions (20% more) missed their deadlines. A certain amount of location error was tolerated by the system because the percentage of firm transactions missing their deadlines depends not only on the accuracy of the LMH location information, but also on the LMH workload, provided that other parameters remain unchanged.

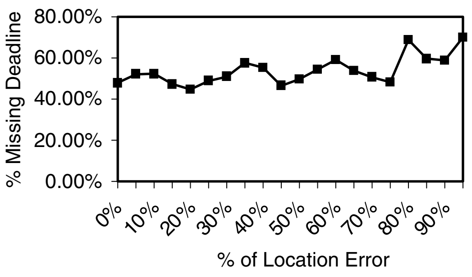

Fig. 11 Effects of location error rate on % firm transactions missing deadlines

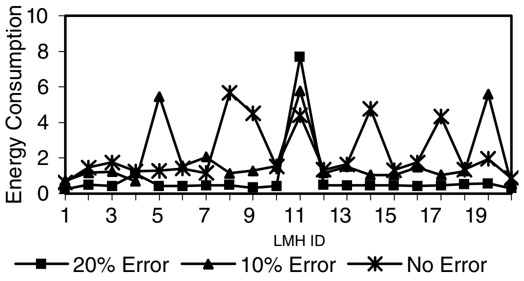

Fig. 12 Effect of error rate in energy information on energy consumption of each LMH

6.5 Varying energy error rate

In this experiment, we have varied the percentage of error in the energy information of each LMH, which is periodically broadcast to the SMHs and other LMHs. We have conducted this experiment with soft transactions as the SMHs will be using the energy information of LMHs to manage soft transactions only. Figure 12 shows that when the energy error rate is low (10% or below), the energy consumption among the LMH is almost balanced. As the energy error rate increases the uniformity in energy consumption among LMHs decreases. This is because of the fact that as the energy error rate increases, the SMHs have a higher chance to identify a wrong LMH as the one with the highest energy (in this experiment, LMH 11 happened to be identified by SMHs as the one which has the highest energy, and thus, it must process more transactions, and subsequently consumed a lot more energy than the other LMHs did). The average difference in energy consumption among LMHs is almost 80% higher when there is 20% error in energy information. So the energy balance is very sensitive to the accuracy of the energy information.

6.6 Varying the speed of mobile hosts

The goal of this set of experiments is to study the effects of mobility of MHs in terms of speed, which is equal to the distance the MHs traveled divided by the total time they took to travel. Initially the moving directions of all MHs are randomly generated from the set of eight possible directions ($\nearrow$, $\swarrow$, $\nwarrow$, $\searrow$, $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$). Regardless of its speed, an MH always moves in the same direction as its initial direction. A MH is considered moving out of the network area (i.e. 1000×1000) if its location exceeds the boundary. In that case, the MH gets disconnected for a randomly generated time

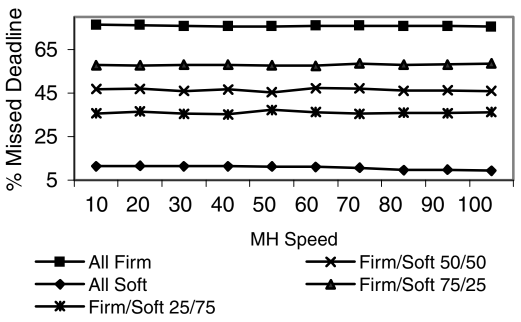

Fig. 13 Effects of MH speed on % missed deadlines

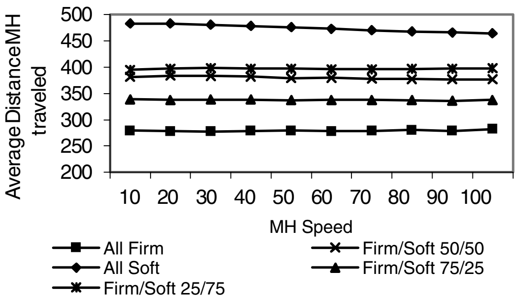

Fig. 14 Effects of MH speed on average distance MHs traveled

period, and reconnected again after the time period by moving in the opposite direction of the last move.

Figure 13 shows that regardless of MH speed, the more firm transactions the system has, the higher percentage of transactions missing deadlines the system yields. This is expected since firm transactions are aborted if they missed their deadlines while soft transactions are aborted only if they missed their second deadlines. The results also show that changing the speed of mobile hosts has negligible effects on the percentage of transaction missing deadlines. This is because of the fact that as MH moves in random directions, some MHs may move closer to each other while some may move further away from each other. As a result, the average distance between MHs that a transaction traveled remains almost constant (Fig. 14). The distance traveled is lowest when all transactions are firm because firm transactions, once missed their deadlines, will not be continued for processing. Since the percentage of transaction missing deadline shows no significant variation when the the speed of MHs changes, the energy consumed by the mobile hosts also shows no significant change as confirmed through our experiments.

7 Conclusions and future work

In this paper, we have introduced a real-time mobile ad-hoc database architecture that can be used to handle applications in battlefields and disaster recovery situations. We have addressed the new issues, i.e., time constraints, energy limitation, and mobility of servers and users related to transaction management for this environment. We have provided a solution for transaction management considering these issues. Our solution is aimed at reducing the percentage of transactions missing deadlines for firm transactions while trying to balance the energy consumption of the servers for soft transactions.

We have provided three alternatives to assign servers to handle transactions initiated by clients. To address timing constraints, our technique handles both soft and firm real-time transactions, reduces the number of firm transactions missing their deadlines by sending firm transactions initiated by an SMH to the LMH that is physically closest to the SMH to process. Our solution also provides a real-time transaction scheduling algorithm that takes transaction types, transaction deadlines and probability of MH disconnection into consideration. To address the energy limitation issue, our technique includes all three modes of MH energy: active, doze, and sleep. It reduces energy consumption by not blindly waiting for an MH to respond to a request but including a formula to compute the time that an SMH should wait for an LMH to respond. To reduce the chance that an MH may get disconnected from the network due to power exhaustion, our technique attempts to achieve a balance in energy consumption among LMHs by sending soft transactions to the LMHs that have the highest remaining energy to process. Our solution also uses vital and non-vital sub-transaction concepts to handle long-lived transactions, uses semantic atomicity to commit compensatable transactions and adds a pre-commit stage to the commit protocol to handle non-compensatable transactions.

We have conducted extensive simulation experiments to analyze the performance of our solution. We have studied the effects of some major parameters by varying their values over a wide range of values in our simulation experiments; this allows us to observe the performance trends of our proposed solution when the parameter values change as they would when the solution is to be applied to an actual system.

From the simulation results, it can be observed that there are some parameters that have significant impacts on the performance of our solution. These impacts are summarized as follows.

(a) When the system needs to reduce the number of transactions missing deadlines only, then Location Based Server Assignment (LBSA) is the best technique. When the system only requires a good balance among the energy consumption of LMHs, then Energy Based Server Assignment (EBSA) is the best technique. But when the system needs to reduce the percentage of transactions missing deadlines as well as to balance the energy consumption among the LMHs, then Transaction Type Based Server Assignment (TTBSA) should be the choice.
(b) The percentage of transactions missing deadlines increases either when the number of mobile hosts decreases or when the system load measured by the transaction arrival rate increases.
(c) The total energy consumption by LMHs increases when the number of LMHs increases. However, when the number of LMHs becomes very high, unless there are only firm transactions in the system, the total energy consumed by LMHs decreases.
(d) The system load also influences the balance in energy consumption distribution in the system for soft transactions. When the system load is light, the energy consumption is uniformly distributed among the LMHs, and thus a better balance of energy consumption is achieved.

(e) The system can tolerate some error in the LMH location information unless the error rate is extremely high. The balance in energy consumption of the system is very sensitive to the accuracy of the LMH energy information.
(f) When mobile hosts move randomly all at the same speed in restricted directions, their speeds have very little impact on the overall performance of the system.

Besides those reported in Section 6, we have also performed additional experiments in which the radius of influence of MHs is a random value in the range of [100, 200] for LMHs and in the range of [50, 100] for SMHs, and the CPU power of MHs is for more modern systems than the ones listed in Table 2. We have adopted the CPU power values from [5] where 1700 MIPS is for LMHs and 100 MIPS for SMHs. Even though the absolute results obtained are different from those reported in Section 6 (for example, when more CPU power was used, fewer transactions missed their deadlines), the relative performances of all the techniques studied remain unchanged. Thus, the same conclusions (a–f) hold when applying our solution to the applications in which more powerful LMHs/SMHs are used and/or MHs do not follow a uniform radius of influence.

In this study, we did not include the energy consumed during the sleep mode in our simulation model because we observed that in all our simulation experiments, with 1000 transactions being simulated, the time the transactions spent in active mode of MHs is much more than the time they spent in doze mode or sleep mode of MHs. Therefore the inclusion of the energy consumption during the sleep mode would not change the overall simulation results.

As an extension of this research work, whether energy consumption for processing transactions can be further reduced should be examined. The PGSG Algorithm [11], which we have used to verify the A/I properties of global transactions, is expensive in terms of time and energy for an ad-hoc environment because during the execution of the algorithm, the coordinator LMH has to maintain connectivity with the participant LMHs. Another disadvantage of the PGSG algorithm is that it does not take care of non-compensatable transactions, real-time transactions, and energy limitation of MHs. So future research can investigate whether this algorithm can be expanded for our environment considering the issues or a new technique is required to verify the A/I properties of global transactions.

Our proposed scheduling algorithm was based on the Least Slack (LS) technique. This technique requires transaction runtime estimates, which is difficult to obtain accurately in a highly dynamic environment. Our future work will examine the Earliest Deadline technique [2].

Future research will also investigate the effects of caching data at SMHs.

References

1. R. Abbott and H. Garcia-Molina, "Scheduling real time transactions," SIGMOD RECORD, vol. 17, no. 1, pp. 71–80, 1988.
2. R. Abbott and H. Garcia-Molina, "Scheduling real time transactions: A performance evaluation," ACM Transactions on Database Systems, vol. 17, no. 3, pp. 1–12, 1992.
3. S. Bandyopadhyay and K. Paul, "Evaluating the performance of mobile agent-based communication among mobile hosts in large ad-hoc wireless network," in the 2nd ACM International Work-

shop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 1999, pp. 69–73.
4. D. Barbara and T. Imielinski, "Sleepers and workaholics: Caching strategies in mobile environments," ACM SIGMOD, pp. 1–12, 1994.
5. M. Brain, "How Microprocessors Work," Howstuffworks Inc., 2002.
6. B.Y. Chan et al., "Cache management for mobile databases: design and evaluation," in Proceedings of 14th International Conference on Data Engineering, 1998, pp. 54–63.
7. P.K. Chrysanthis, "Transaction processing in mobile computing environments," IEEE Workshop on Advances in Parallel and Distributed Systems, 1993, pp. 77–82.
8. DECdirect Workgroup Solutions Catalog, Winter, 1993.
9. R. Dirckze and L. Gruenwald, "A toggle transaction management technique for mobile multidatabases," ACM Conference on Information and Knowledge Management, 1998, pp. 371–377.
10. R. Dirckze, "Transaction management in mobile multi-databases," PhD Dissertation, School of Computer Science, University of Oklahoma, Dec. 1999.
11. R. Dirckze and L. Gruenwald, "A pre-serialization transaction management technique for mobile multi-databases," Special Issue on Software Architecture for Mobile Applications, MONET 2000, pp. 311–321.
12. M.H. Dunham, A. Helal, and S. Balakrishnan, "A mobile transaction model that captures both the data and movement behavior," Mobile Network and Applications, vol. 2, no. 2, pp. 149–162, 1997.
13. J. Gray and A. Reuter, Transaction Processing: Concepts and Techniques, Morgan Kaufmann Publishers, Inc., 1993.
14. L. Gruenwald and S. Banik, "A power-aware technique to manage real-time database transactions in mobile ad-hoc networks," in 4th International Workshop on Mobility in Database and Distributed Systems, part of the International Conference on Database and EXpert systems Applications (DEXA), 2001, pp. 570–574.
15. X. Hong, M. Gerla, R. Bagrodia, and G. Pei, "A group mobility model for ad-hoc wireless networks," in Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 1999, pp. 53–60.
16. T. Imielinski and B.R. Badrinath, "Mobile wireless computing: Solutions and challenges in data management," Communications of the ACM (CACM), vol. 37, pp. 18–28, 1994.
17. B. Kao and H. Garcia-Molina, "Deadline assignment in a distributed soft real-time systems," in Proceedings of the 13th International Conference on Distributed Computing Systems, 1993., pp. 428–437.
18. E. Kayan and O. Ulusoy, "Real-time transaction management in mobile computing systems," in 6th International Conference on Database Systems for Advanced Applications, 1999, pp. 127–134.
19. D.W. Kelton, R.P. Sadowski, and D.A. Sadowski, Simulation with Arena, WCB/McGraw-Hill Publishers, 1998.
20. Y.-B. Ko and N.H. Vaidya, "Location-aided routing (LAR) in mobile Ad-Hoc networks," MOBICOM, pp. 66–75, 1998.
21. P. Krishna, N.H. Vaidya, and D.K. Pradhan, "Location management in distributed mobile environments," in Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems, 1994, pp. 81–88.
22. V.C.S. Lee et al., "Real-time transactions processing with partial validation at mobile clients," in Proceedings of 7th International Conference on Real-time Computing Systems and Applications, 2000, pp. 473–477.
23. E. Levy, H.F. Korth, and A. Silberschatz, "An optimistic commit protocol for distributed transaction management," in Proceedings of ACM-SGMOD International Conference on Management of Data, Colorado, 1991, pp. 88–97.
24. J.B. Lim and A.R. Hurson, "Transaction processing in mobile heterogeneous database systems," IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 8, pp. 1330–1346, 2002.
25. M. Liu, J.S. Baras, S.M. Payne, and H. Harrelson, "Modeling and simulation of large hybrid networks," in Proceeding of 2nd Annual Advanced Telecommunications/Infrastructure Distribution Research Program (ATIRP) Conference, 1999, pp. 1–12.
26. S.K. Madria and B. Bhargava, "A transaction model for mobile computing," International Database Engineering and Application Symposium (IDEAS 1998), 1998, pp. 92–102.
27. S.K. Madria, M. Baseer, and S.S. Bhowmick, "A multi-version transaction model to improve data availability in mobile computing," CoopIS/DOA/ODBASE 2002, vol. LNCS 2519, pp. 322–338.
28. C. Michelle et al., "Aspects of energy conservation on the St. George University of Toronto Campus," A report by Division of Environment at the University of Toronto, 1996/97.

(http://www.cquest.utoronto.ca/env/env421h/energy).
29. T.M. Ozsu and P. Valduriez, Principles of Distributed Database Systems, Prentice Hall, Englewood Cliff, N.J., 1991.
30. E. Pitoura and B. Bhargava, "A framework for providing consistent and recoverable agent-based access to heterogeneous mobile databases," SIGMOD Record, pp. 44–49, 1995.
31. A. Alan, B. Pritsker, and J.J. O'Reilly, Simulation with Visual SLAM and AweSim, Systems Publishing Corporation, 1999.
32. K. Ramamritham, "Real-time databases," Distributed and Parallel Databases, vol. 1, no. 2, pp. 199–226, April 1993.
33. T. Kian-Lee, C. Jun, and O.B. Chin, "An evaluation of cache invalidation strategies in wireless environments," The IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 8, pp. 789–807, 2001.
34. J. Warren, T. Martin, A. Smailagic, and D.P. Siewiorek, "System design approach to power aware mobile computers," in Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'03), Florida, 2003.
35. L.H. Yeo and A. Zaslavsky, "Submission of transactions from mobile workstations in a cooperative MDB processing environment," in 14th International Conference on Distributed Computing Systems, Poland, 1994, pp. 372–379.
36. J.M. DiMicco, "Mobile ad hoc voting," http://web.media.mit.edu/∼joanie/voting/mobile-adhoc-voting.pdf.
37. J. Griffioen, W.B. Seales, and J.E. Lumpp, "Teaching in realtime wireless classrooms," The 1998 Frontiers in Education Conference, Nov. 1998.
38. V. Kanitkar and A. Delis, "Efficient processing of client transactions in real-time," Distributed and Parallel Databases, vol. 17, no. 1, pp. 39–74, 2005.