A Power-Aware Technique to Manage Real-Time Database Transactions in Mobile Ad-Hoc Networks [**]

Le Gruenwald , Shankar M. Banik
University of Oklahoma
School of Computer Science
Norman, OK 73019
(ggruenwald@ou.edu; smbanik@ou.edu)

Abstract

A mobile ad-hoc network (MANET) is a collection of wireless autonomous mobile hosts which are free to move randomly, thus forming a temporary network without any fixed backbone infrastructure. MANET is typically used in battlefields and disaster recovery situations where temporary network connectivity is required. 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 presents a performance study by means of simulation.

1. Introduction

Transaction Manager (TM) of a mobile multidatabase management system is responsible for providing reliable and consistent units of computing to its users. 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 migrates to a new cell, it is under the control of the new cell's MSS. 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 fixed MSSs. MANET is widely used in battlefields and in disaster recovery situations ([7][10]).

Much research in the area of mobile database transaction management was based on the first architecture ([15], [12], [6], [11], [5]), while none on the second one. Supporting database transaction services in MANET 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 applications in this environment are time-critical which require their transactions to be executed not only correctly but also within their deadlines. Thus the TM 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 MANET 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 the TM should also provides 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 can be reduced.

This paper presents a transaction management solution that takes ad-hoc networks, real-time constraints, and energy efficiency into consideration. The rest of the paper is organized as follows. Sections 2 and 3 describe the proposed MANET architecture and transaction management solution. Section 4 presents the simulation results. Finally Section 5 concludes the paper with future research.

2. Proposed Architecture

Depending on communication capacity, computing power, disk storage, size of memory and energy

[**] This work is partially supported by the National Science Foundation grant No. EIA-9973465

limitation, MHs can be classified into two groups: 1) computers with reduced memory, storage, power and computing capabilities, which we will call *Small Mobile Hosts* (SMH*s*), and 2) classical workstations equipped with more storage, power, communication and computing facilities than the SMH*s*, which we will call *Large Mobile Host* (LMH*s*). Every MH has a radius of influence. An MH can directly communicate with other MHs, which are within its radius of influence. 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 intermediate MH*s* between them [2].

To reduce energy consumption, an MH can operate in three modes - Active mode, Doze mode and Sleep mode. In the active mode, the MH performs its usual activities, its CPU is working and its communication device can transmit and receive signals. In the doze mode, the CPU of the MH will be working on a lower rate but it can examine messages from other MHs, and the communication device can receive signals in this mode; so the MH can be awaken by a message from other MHs [3]. In the sleep mode, both the CPU and the communication device of the MH are suspended. Due to energy and storage limitations, we will assume that only LMH*s* will store the whole DBMS and SMH*s* will store only some modules of the DBMS (e.g. Query Processor) that allow them to query their own data, submit transactions to LMH*s* and receive the results. The DBMSs at LMHs can be heterogeneous.

This proposed architecture can be used to support many applications, such as battlefields and disaster recovery. In battlefields, portable computing device with soldiers will work as SMH*s* while computers stored in tanks and humvees will work as LMH*s*. These LMH*s* can store tactical information regarding enemy and other units in a database and the SMH*s* can communicate with the LMH*s* to get information from the database (e.g. which unit of enemy is located where, what is their strength). In a disaster recovery operation, the palmtops carried by rescuers can be viewed as the SMH*s* and the computers in mobile hospitals can be viewed as the LMH*s*. The LMH*s* can keep the information of all the medical equipments in their databases, and the SMH*s* can query about the inventory and inform the LMH*s* to keep ready a certain arrangement for a particular patient whom they have found in their site.

3. Proposed Solution

3.1. Key Information Stored at Mobile Hosts

Each MH will store some key information in its local database. The *ID* field uniquely identifies an MH. Every MH will get its coordinates from GPS [9] periodically and store them in the *Position* field to be used at the time of routing. The *Energy_level* field records the amount of energy available at that time. Each LMH also maintains a *Global Schem*a - an integration of all local schemas from all LMH*s*, and the *ID* of the LMH for each local schema. This *Global Schema* is required to identify which data object is stored in which LMH. Each LMH will periodically broadcast its *ID*, *Position* and *Enery_level,* which the SMHs and other LMH*s* will record in their local databases after listening to the broadcast channel.

3.2. Transaction Properties and Classification

In our real-time environment, transactions have deadlines and are classified into two categories: firm and soft. Firm transactions must be aborted if they miss their deadlines while soft transactions still can be executed after their deadlines have expired. The value of a firm transaction becomes zero after its deadline expires [14]. From the value function describing tasks with soft deadlines in [1], we can define soft transactions with two deadlines: it still can be executed after its first deadline expires, but its value decreases after the first deadline and becomes zero after the second deadline. A global transaction may consist of a number of sub-transactions running on other LMHs. A sub-transaction is either vital or non-vital [4]. All the vital sub-transactions of a global transaction must succeed in order for it to succeed.

3.3. Overview of the Proposed Transaction Management Technique

We assume that when an SMH initiates a transaction, it will send its entire transaction to an LMH to process. There are two parts of transaction management: how an SMH submits a transaction to an LMH and how an LMH executes that transaction and returns its result to the SMH. We address energy limitation by using three MH energy modes (active, sleep, and doze) at different stages of transaction execution. We also provide a balance in LMH energy consumption by executing soft transactions at the LMH with the highest energy level. To address the real-time behavior of transactions, we reduce the number of transactions missing deadlines by executing firm transactions at the nearest LMH, employing two deadlines for soft transactions, and scheduling both soft and firm transactions using a real-time energy-efficient transaction-scheduling algorithm. To address disconnection and migration that cause prolonged execution of mobile transactions and disconnection due to catastrophic failures, we introduce the concepts of toggled transactions and suspended transactions that allow a transaction to be verified for its violation of the Atomicity and Isolation properties as soon as its vital sub-transactions are completed, and allow disconnected transactions to wait in the system unless they obstruct the execution of another transaction.

3.3.1. Energy-Efficient Real-Time Transaction Scheduling Algorithm

Our scheduling algorithm considers not only transaction types (firm and soft), transaction deadlines, but also MH energy limitation. Here we have modified the Least Slack (LS) cognizant technique proposed in [1] with respect to energy constraints, disconnections and transaction types. We calculate a transaction's slack time s as follows: $s = d - (t + c + Pd*Td)$ where d is the deadline, t the current time, c the runtime estimate, Pd the probability of disconnection during execution and Td the average time loss due to disconnection. We sort all the transactions with respect to their slack times irrespective of their transaction types and give higher priorities to transactions that have shorter slack times except for the following cases. If two firm transactions have the same slack time, then a higher priority will be given to the one whose requester has less energy level because the requester which has less energy will exhaust its energy earlier, it is better to schedule its transactions earlier. We assume that MHs while submitting their transactions/sub-transactions to an LMH will also send their energy levels to the LMH. The same technique 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. If the slack time of a soft transaction is found to be negative, then its slack time will be recalculated considering 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.

3.3.2. Transaction Submission from SMH to LMH

In our transaction management solution, an SMH will submit its firm transactions to the nearest LMH and its soft transactions to the highest energy level LMH. Now if the nearest LMH is in active mode, it will receive and process the transaction. If it is in doze mode and the transaction is firm, it will wake up, start processing the transaction. But if the LMH is in sleep mode, it will not be able to receive the transaction; the requesting SMH will then wait for some time period. If the SMH 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 length of the time period using the transaction run time estimate, communication overhead and possible delay due to disconnection. If the transaction is soft, the SMH will find the LMH with the highest energy level from its local database, find a route to this LMH and submit the transaction to it. The SMH, after getting the result of the transaction, will send an acknowledgement to the LMH which has processed the global transaction.

3.3.3. Transaction Processing at LMH and Result Submission to SMH

After receiving a transaction from an SMH, if the LMH is in active mode, the Transaction Scheduler (TS) of the LMH will schedule the transaction according to real-time scheduling algorithm described in Section 3.3.1. The Transaction Coordinator (TC) of the LMH will find the participant LMH*s* which contain the required data items for this transaction after consulting the *Global Schema*. Then it will divide the global transaction into sub-transactions such that all data required by a sub-transaction resides at one single participant LMH. The TC will then distribute the deadline of the global transaction among the sub-transactions using the EQF Strategy proposed in [8]. Then the TC will find the routes to LMH*s* and submit the corresponding sub-transactions to them.

When all the vital sub-transactions of a global transaction are completed, the TC will run the Partial Global Serialization Graph (PGSG) algorithm that we have developed for the General Mobile Computing Architecture [5] to verify whether the global transaction has violated the Atomicity/Isolation (A/I) properties. If all vital sub-transactions can commit, then the global transaction does not violate the Atomicity property. Then the PGSG algorithm with check for the Isolation property violation using the concept of serialization graphs.

If the A/I properties are not violated, then the transaction is *toggled* and its execution continues until all its remaining non-vital sub-transactions are either committed or aborted. LMH can then commit the transaction and submit the result to the requesting SMH. Note that 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. A transaction is said to be in a *suspended* state (i.e. its transaction is halted, no new sub-transaction can be initiated) if its MH is disconnected from the network and the system determines that this MH has encountered a catastrophic failure.

But if the A/I properties are violated and the transaction has not yet missed the deadline (or the second deadline for a soft transaction), the LMH will restart the transaction; otherwise the LMH will abort the transaction. After the TC has decided to commit/abort a transaction, it will find a route to the requesting SMH for submitting the result of the transaction. If the requesting SMH is in active mode, it will receive the result and send an acknowledgement. If it is in doze mode and the transaction is firm, then it will wake up, receive the result and send an acknowledgement. But if the transaction is soft, SMH will calculate the remaining slack time for the transaction. The SMH will wait until the slack time is less than some time period value, then it will wake up, receive the result and send an acknowledgement. If the requesting

SMH is in sleep mode, it will not be able to receive the result. So if the TC of the LMH which has processed the transaction does not receive any acknowledgement till the deadline of the transaction, it will assume that the requesting SMH is in sleep mode. If the transaction is firm, it will abort the transaction. But if the transaction is soft, 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 lot 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 will depend on the remaining slack time of the transaction and the energy level of the LMH. If the energy level is low, the interval will be large and the number of transmissions will be small.

4. Simulation Results

We study the performance of our proposed technique by means of simulation. The simulation model is implemented using the Awesim simulation tool [13]. Global transactions are defined as entities and mobile hosts are defined as resources with identical initial energy levels and randomly distributed locations. We measure the performance in terms of percentage of transactions missing deadlines and amount of energy consumption of each resource, which is equal to the power of the resource multiplied with the time the resource was in active mode.

In the first experiment, we have varied the number of LMH*s* for different mixtures of firm and soft transactions. Figure 1 shows that more 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 will need less waiting time to use the servers and thus fewer transactions will miss their deadlines. The total energy consumption in LMHs increases as the number of LMHs increases [Figure 2]. 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.

In the second experiment, we have varied the inter-arrival time (IAT) of soft transactions and calculated the energy consumption of each LMH to see the effect of the system load on energy consumption distribution among individual LMHs. 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 the 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 level. That means if the IAT of transactions is small, then all the SMH*s* may not have the recent values of energy levels of all LMH*s*. As a result, they will not be able to correctly identify the LMH with the highest energy level for soft transactions. So the total energy consumption in the system will not be evenly distributed among all the LMH*s*. Figure 3 and 4 shows that when the IAT is EXPON(1) energy consumption of LMHs is not uniformly distributed, but when the IAT increases to EXPON(5), the energy consumption of LMHs are almost uniform.

5. Conclusions

In this paper, we have introduced a mobile ad-hoc network database architecture that can be used to support applications such as battlefields and disaster recovery operations. We have provided a solution for transaction management considering transaction real-time constraints as well as mobility and energy limitation of both servers and clients. Our solution is aimed at reducing the percentage of transactions missing deadlines while saving the energy consumption by both clients and servers and balancing the energy consumption by servers. Our simulation results indicate the following: 1) when there is a lower system load of soft transactions, a better balance of energy consumption among servers is achieved; 2) the system performs better when there are more servers.

For future research, we plan to examine other alternatives for managing firm/soft transactions, such as sending all transactions to nearest servers only, to highest energy level servers only, or to the randomly selected servers only. We will study the impact of roaming frequencies of both servers and clients.

6. References

[1] Abbott R., H. Garcia-Molina, "Scheduling Real Time Transactions", SIGMOD RECORD, Vol. 17, No. 1, March 1988.

[2] Bandyopadhyay, S., and K. Paul, "Evaluating the Performance of Mobile Agent-Based Communication among Mobile Hosts in Large Ad-Hoc Wireless Network", MSWIM 1999.

[3] Barbara D., T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments", ACM SIGMOD, May 1994.

[4] Chrysanthis, P.K., "Transaction Processing in Mobile Computing Environments", IEEE Workshop on Advances in Parallel and Distributed Systems, October 1993.

[5] Dirckze, R. and L. Gruenwald, "A Pre-serialization Transaction Management Technique for Mobile Multi-databases ", Special Issue on Software Architecture for Mobile Applications, Vol.5, No.4, 2000.

[6] Dunham M., A. Helal, S. Balakrishnan S., "A Mobile Transaction Model that Captures Both the Data and Movement Behavior", Mobile Network and Applications, Vol. 2, No. 2, October 1997.

[7] Hong X., et al, "A Group Mobility Model for Ad Hoc Wireless Networks", MSWIM, 1999.

[8] Kao B., H. Garcia-Molina, "Deadline Assignment in a Distributed Soft Real-Time Systems", Proceedings of the 13th International Conference on Distributed Computing Systems. May 1993.

[9] Ko, Y., N. Vaidya , "Location-Aided Routing (LAR) in Mobile Ad-Hoc Networks", MOBICOM 1998.

[10] Liu, M., et al., "Modeling and Simulation of large Hybrid Networks", Proceeding of 2nd Annual Advanced Telecommunications/ Infrastructure Distribution Research Program (ATIRP) Conference 1999.

[11] Madria, S. K., B. K. Bhargava, "A Transaction Model for Mobile Computing", International Database Engineering and Application Symposium (IDEAS 1998), July 1998.

[12] Pitoura, S. K., B. K. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environment", 15th Int. Conference on Distributed Computing System, June 1995.

[13] Pritsker A. Alan B, O'Reilly Jean J., "Simulation with Visual SLAM and Awesim", Systems Publishing Corporation, 1999.

[14] Ramamritham K., "Real-Time Databases", Distributed and Parallel Databases, Vol. 1, No. 2, April 1993, pp 199-226.

[15] Walborn G. D., P. K. Chrysanthis, "Supporting Semantic-Based Transaction Processing in Mobile Database Applications", 14th IEEE Symposium on Reliable Distributed Systems, Sept. 1994.

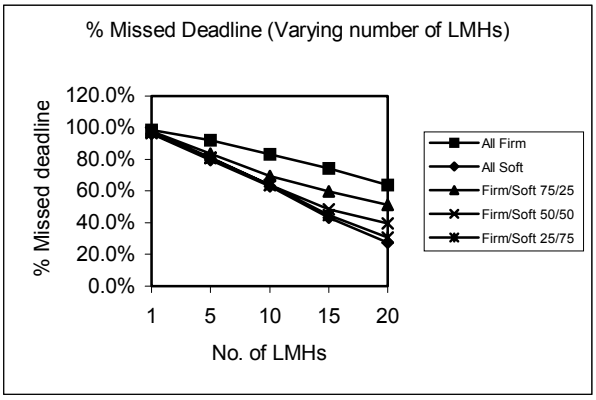

Figure 1. Impact of No. of LMHs on % Missed Deadline

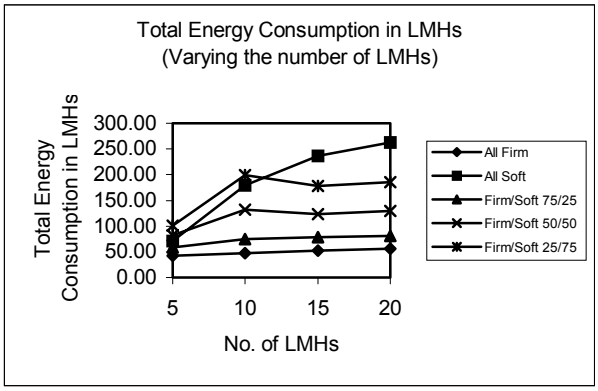

Figure 2. Impact of No. of LMHs on Total Energy Consumption

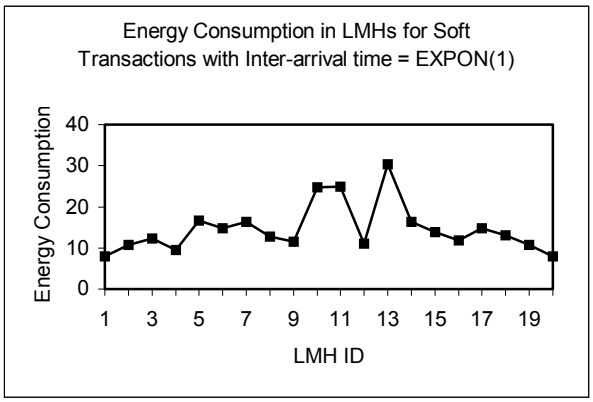

Figure 3. Impact of IAT on Energy Consumption of LMHs for soft transactions

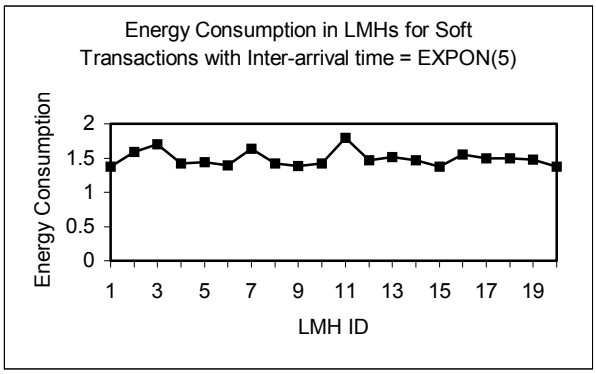

Figure 4. Impact of IAT on Energy Consumption of LMHs for soft transactions