Energy-Efficient Transaction Management for Mobile Ad-hoc Network Databases

1st Author
1st author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
1st author's email address

2nd Author
2nd author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
2nd author's email address

ABSTRACT
A Mobile Ad-hoc Network (MANET) is a collection of mobile, wireless and battery-powered nodes without a fixed infrastructure; so MANET fits well in disaster rescue and military operations. However, when a node runs out of power or has insufficient power to function, communication may fail, disconnections may happen, execution of transactions may be prolonged, and some transactions may be aborted if they are time-critical and miss their deadlines. In this paper, we propose an energy-efficient transaction management technique for MANET databases in a clustered network architecture where nodes are divided into clusters, each of which has a node, called cluster head, responsible for the processing of all nodes in the cluster. In our technique, in order to conserve energy and balance the energy consumption among servers so that the lifetime of the network is prolonged, we elect cluster heads to work as coordinating servers, re-elect a cluster head locally for a cluster when its current cluster head's remaining energy is low, and propose an optimistic concurrency control to offer high concurrency and avoid wasting limited system resources. The simulation results confirm that our technique performs better than other existing techniques in terms of transaction abort rate and degree of balancing energy consumption among servers.

Keywords
Mobile ad-hoc network, clustering, transaction management, concurrency control.

1. INTRODUCTION
A mobile database system built on a Mobile Ad-hoc Network (MANET) is called a MANET database system. In this system, both clients and servers are mobile, wireless and battery-powered, and the databases are stored at servers and accessed by clients. As no fixed infrastructure is required, MANET databases can be deployed in a short time and mobile users can access and manipulate data anytime and anywhere, and they become an attractive solution to handle mission-critical database applications, such as disaster response and recovery system [4, 12, 1] and military operations like battlefields [1]. In these applications, transactions must be executed not only correctly but also within their deadlines. To fulfill this, a transaction management technique is essential.

Database management systems ensure convenient and efficient access of databases for their users, and transaction Manager (TM) is a vital component in the system. TM is responsible for ensuring that a database remains in a correct/consistent state even if system fails. Also TM ensures that concurrent transaction executions proceed without interleaving [14]. However, because of the mobility and portability, mobile nodes have severe resource constraints in terms of capacity of battery, memory size and CPU speed [8]. As the battery capacity is limited, it compromises the ability of each mobile node to support services and applications [6]. Also the battery technology is not developed as rapidly as the mobile devices and wireless technologies, so that the limited battery lifetime is always a bottleneck for the development of improved mobile devices [16]. Therefore, a suitable transaction management for MANET databases should be energy-efficient. Here, energy efficiency refers to the amount of service work that a system can accomplish when its battery capacity is limited [8].

Several transaction management techniques for MANET databases have been proposed; however, either most of them do not address the energy efficiency issue [3, 13], or other handles the energy efficiency issue well but they assume that all transactions are compensable (it is not true for mission-critical applications) in their approach [10]. To fill in this gap, in this paper we propose a transaction management solution that takes mobility, real-time constraints and energy efficiency into considerations, and is specially designed for mission-critical MANET databases. Our object is to minimize energy consumption of each mobile node and balance energy consumption among mobile nodes, so that mobile nodes with low energy do not run out of energy quickly, and thus, the number of disconnections and transaction aborts due to low energy or energy exhaustion can be reduced. 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. Section 4 presents our transaction management solution. Section 5 presents the simulation results. Finally Section 6 concludes the paper with future research.

Conference'10, Month 1–2, 2010, City, State, Country.

2. RELATED WORK

In this section, first we briefly review proposed transaction management techniques for MANET databases, and then we summarize them in terms of drawbacks.

Semantic Serializability Applied to Mobility (SESAMO) [3] is proposed for MANET databases. SESAMO does not require that a single mobile node plays the role of the centralized coordinator for all global transactions. Moreover, SESAMO allows that sub-transactions commit independently of the commitment of their respective global transactions. SESAMO is based on the semantic serializability, which assumes that databases are disjoint and updates on a database only depend on values of data of the same database; therefore, the atomicity and global serializability can be relaxed. However, in SESAMO, global transactions still need be serialized at each site using strict 2PL globally, while at the same time, each site maintains the consistency of its own local database.

A transaction management for real-time MANET databases is proposed in [10]. To reduce energy consumption, each mobile node operates in three modes: active, doze, and sleep; a client does not blindly wait for a server to respond to a request, but a formula (sum of runtime estimate of the transaction, communication time and delay time due to disconnection) is introduced to compute the time that a client should wait for a server to respond. To reduce the chance that a node may get disconnected from the network due to power exhaustion, soft transactions are sent to the servers that have the highest remaining energy to process so that energy consumption among servers can be balanced. The Partial Global Serialization Graph (PGSG) technique proposed in [7] is adopted to guarantee transaction atomicity and isolation and PGSG assumes that all transactions are compensable.

In [13], the authors only focus on the problem of setting up reasonable time-outs when guaranteeing atomicity for transaction processing within MANETs and propose the Adjourn State, which does not rely on time-outs and is a non-blocking state. That is, Adjourn State allows concurrent transactions to be processed while a distributed transaction is waiting for the Coordinator's vote message to demand the commit decision. Also, the Commit tree, a tree data structure that represents the execution status of all active sub-transactions, is introduced to efficiently coordinate transactions.

Although the solution in [10] handles the energy problem well, PGSG requires that every transaction is compensable and compensable transactions are not always available in mission-critical applications. The approach in [13] mainly focuses on transaction atomicity and does not address the energy problem. SESAMO does not address the energy problem either, and it assumes that a MANET database is seen as a collection of semantic units, which are disjoint and do not depend on each other; therefore the atomicity and global serializability of global transactions can be relaxed [3]. However, the MANET databases for mission-critical applications cannot relax the atomicity and global serializability for their transactions because each database depends on each other such that semantic units cannot be defined. For example, in a disaster rescue scenario, before sending firefighters to pursue some actions, the status of their equipment has to be checked, where the firefighter database table may be stored on one mobile server, and the equipment database table may be stored on another mobile server. In a battlefield scene, before a tank fires cannon, the locations of the soldiers have to be checked to ensure their safety, where the tank database table is stored on one mobile server, and the soldiers' information is located on another mobile server.

3. PROPOSED ARCHITECTURE

In this section, we introduce our proposed architecture: a clustered MANET, which is built by applying our robust weighted clustering algorithm called PMW (Power, Mobility and Workload) [18]. In the architecture as shown in Figure 1, mobile nodes are divided into clusters, and each cluster has one cluster head working as coordinating server to be responsible for the transaction processing within the cluster. We choose this architecture because of two reasons. First, many MANET applications in the literature use grouped or clustered system architectures: disaster response and recovery system using MANETs [4, 12], mobile telemedicine system [16], and mobile group-based multimedia cooperation system [9]. Second, because every node is mobile in MANETs, the network topology may change rapidly and unpredictably over time. According to [5], clustered architectures are proper to keep the network topology stable as long as possible, so that the performance of routing and resource relocation protocols is not compromised.

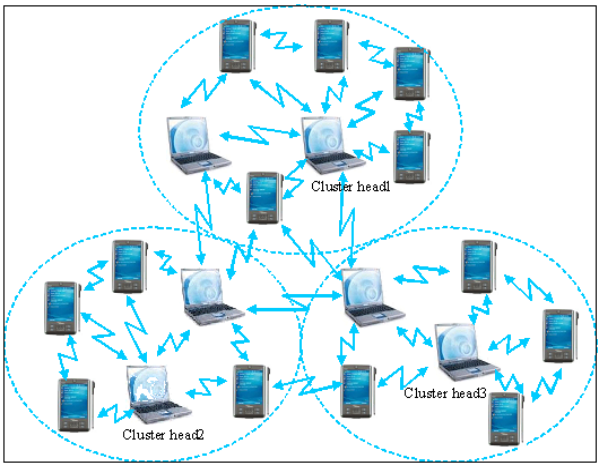

Figure 1. Architecture of a clustered MANET database

3.1 Classification of Mobile Nodes from Database Perspective and Database Partition

In our clustered MANET architecture, depending on the communication strength, computing power and storage size, mobile nodes are classified into either clients or servers. On clients, only the query processing modules that allow them to submit transactions and receive results are installed; while on servers, the complete database management system are installed and servers provide transaction processing services. Servers are further classified into coordinating servers (actually replaced by cluster heads to solve energy efficiency issue) or participating servers. Coordinating servers are the ones which receive global transactions, divide them into sub-transactions, forward these sub-transactions to appropriate participating servers, and maintain the ACID (Atomicity, Consistency, Isolation, and Durability)

properties of global transactions. Participating servers are the ones that process sub-transactions transmitted from coordinating servers and local transactions submitted from local users, and preserve their ACID properties. Coordinating and participating servers are not necessarily physical units, and one server can function as both.

The entire database is partitioned into local databases and distributed at different servers, and there is no caching or replication technique involved for simplicity. Transactions are based on the simple flat model, which contains a set of read, write, insert, and delete operations. However, when operations are checked for conflicts, insert and delete operations are treated just as write ones. Any subset of operations of a global transaction that access the same server is submitted and executed as a single sub-transaction [7].

3.2 Classification of Mobile Nodes from Network Perspective

A clustered MANET consists of cluster heads and cluster members, where a cluster head (like a mobile support station in a cellular mobile network) manages its clusters, coordinates intra/inter-cluster communication and so on. A cluster member is a node that belongs to a cluster and is not a cluster head.

3.3 Electing Stable Cluster Head based on Nodes' Power, Mobility and Workload

To capture the mobility of nodes, we do not consider the absolute roaming speed. This is because it is easy to calculate the speed's quantity but it is hard to predict the direction of movement. Without the direction, the speed's quantity alone is not appropriate to justify whether a node is a good candidate or not. For instance, the speeds of two nodes are small, but they both move in the opposite directions. As time goes, they will be out of each other's transmission range and get disconnected from each other. Also the utilization of GPS is opted out because GPS consumes the limited battery power of mobile nodes.

- Instead, two mobility metrics, Relative Mobility (RM) [2] and Mobility Prediction (MP), are introduced to monitor the mobility of nodes and applied to determine whether a node is suitable to be a cluster head as follows: For each node j ($1 \le j \le N$ for N nodes in the network), after receiving two successive HELLO messages from every 1- hop neighbor i ($1 \le i \le n$ if there are n neighbors), the RM_{ij} is calculated by the Formula (3.1). rss_{ij1} and rss_{ij2} are the received signal strength (RSS) that are read from the RSS indicator when the first and second HELLO message from the same neighbor are received, respectively. Based on the value of RM_{ij}, we can say that if RM_{ij} is equal to 1, then the node j and its neighbor i either do not move at all or move with the same speed in the same direction; if RM_{ij} is less than 1, then they move close to each other; otherwise, they move away from each other.

$$RM_{ij} = \frac{rss_{ij1}}{rss_{ij2}} \tag{3.1}$$

- For each node j, to take into account the mobility of all n 1-hop neighbors, MP_j is calculated as the standard deviation of $RM_{1j}, RM_{2j}, \ldots, RM_{nj}$ shown in the Formula (3.2). However, for the stability of elected clusters, we prefer RM_{ij} to be equal to or less than 1 because we want cluster heads not to move away from their members. Thus, in the MP_j calculation the mean of RM_{ij} ($1 \le i \le n$) is 1 instead of the actual mean. A node j with a lower MP_j means that it stays closer to its neighbors, thus, it is a better candidate for the cluster head among its neighbors.

$$MP_j = \sqrt{\frac{\sum_{i=1}^{n} (RM_{ij} - \overline{RM_{ij}})^2}{n}}, \quad where\ \overline{RM_{ij}} = 1 \tag{3.2}$$

When dealing with the limited battery power, we consider not only the Remaining Power (*RP*) of each node but also its Power Decrease Rate (*PDR*) as the workload because nodes with heavier workload consume more energy, so that we can balance the power usage and prevent cluster heads from running out of power quickly. In other words, for each node j, the PDR_j is considered because the RP_j represents only the current state of power level and the power will run out soon if this node usually has a heavy workload (for instance, it provides service as a server and relays packets for many neighbors). The PDR_j at time interval $[t_1, t_2]$ is calculated by using the Formula (3.3), where rp_{j1} and rp_{j2} are the remaining power at time t_1 and t_2, respectively.

$$PDR_j = \frac{rp_{j_1} - rp_{j_2}}{t_2 - t_1} \tag{3.3}$$

A node with a lower *PDR* indicates that it was not busy at least during the interval $[t_1, t_2]$. However, when a node had a busy work history, it most likely would be busy in the future as well. Since the larger the time interval is, the more accurate the *PDR* is in indicating a node's workload history, during the initial election, each node saves a copy of its initial remaining power and initial time as rp_{j1} and t_1, such that a more accurate *PDR* can be calculated in the future re-election.

Based on the above analysis about power, mobility and workload, it is obvious that a node j is the best candidate for a cluster head among all its neighbors if its RP_j is the highest, its MP_j is the lowest and its PDR_j is the lowest. In other words, a node with the highest weight is the best candidate for a cluster head when we combine these three metrics together as the weight, which is calculated in Formula (3.4). Since these metrics have different units, we apply the inversed exponential function to normalize MP_j and PDR_j and bound their values between 0 and 1. RP_j is left out because it is the remaining power level in percentage and its value is already between 0 and 1.

$$W_j = f_1 * e^{-MP_j} + f_2 * RP_j + f_3 * e^{-PDR_j} \tag{3.4}$$

Due to less communication strength, less computing power and smaller storage size, clients cannot be elected as cluster heads. The simulation results from the NS-2 simulator did confirm that PMW prolonged the lifetime of a clustered MANET and outperformed MOBIC [2] by as much as 23% on average. More details can be found in [18].

4. PROPOSED TRANSACTION MANAGEMENT

In this section, we present our transaction management technique that addresses the above energy problem by answering the following questions: how a transaction is processed in a clustered MANET, how to handle transaction submission from client to server, and how to guarantee the atomicity and isolation properties of transactions.

4.1 Overall Transaction Flow in a Clustered MANET

We assume that when a client initiates a transaction T, T is sent to its cluster head directly; the cluster head divides T into several sub-transactions, and transmits them to the appropriate participating servers according to the global schema. Each participating server processes the sub-transactions locally, and sends the results back to the cluster head. The cluster head runs the 2PC, gathers all results from the participating servers and validates T globally. The overall transaction flow is also shown in Figure 2.

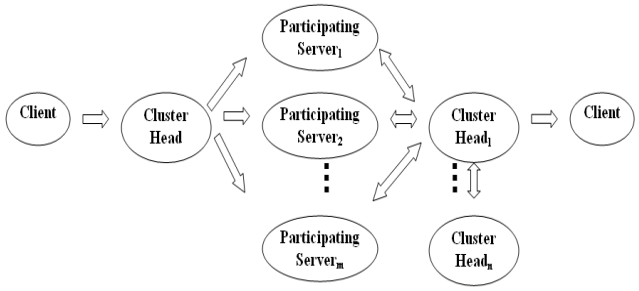

Figure 2. Transaction flow in a clustered MANET

4.2 Handling Transaction Submission from Client to Server

In MANETs, both the clients and servers are mobile, so finding a route from one node to another node is necessary before submitting a transaction. Also in mission-critical MANET databases, since mobile nodes have energy limitation and transactions have timing constraints, a suitable transaction management technique has to balance the energy consumption among servers to reduce the chance that servers get disconnected due to insufficient power or power exhaustion; while at the same time the number of aborted transactions should be reduced due to missing their deadlines. Therefore, clients submit transactions directly to their own cluster heads so that this eliminates the overhead that clients must spend to identify suitable servers to send their transaction to, and the transactions can meet their deadlines.

4.3 Guaranteeing Atomicity and Isolation of Transactions

To verify the isolation property of transactions we adopt the Sequential Order with Dynamic Adjustment (SODA) [17]. In the design of SODA, the transaction real-time constraints as well as mobility and energy limitation of both servers and clients are taken into considerations. SODA is based on optimistic CC to offer high concurrency and avoid unbounded blocking time that may cause many transactions miss their deadlines, utilizes the sequential order of committed transactions to improve response time, and dynamically adjusts sequential order of transactions to reduce transaction aborts. To deal with the atomicity, we simply adopt the well-known 2-phase commit (2PC).

In order to make SODA work effectively in clustered MANET databases, the coordinating server functionality is combined in the cluster head because a cluster head is the nearest server with the highest power in the neighborhood of clients. Therefore, in our transaction management technique, cluster heads and participating servers cooperate together to ensure the atomicity and isolation properties of transaction. The functionalities of cluster heads and participating servers are shown below. Note that one server can have both the functionalities at the same time

4.3.1 The participating server functionality

A participating server has following tasks to accomplish:

- It receives and processes sub-transactions and local transactions.
- When it receives the request about the status of sub-transactions, it runs SODA locally based on the local sequential order of committed transactions.
- It sends the final result to the requesting cluster head. If a sub-transaction passes the validation, the timestamp of read set is also sent to the cluster head.
- If the sub-transaction commits, it rebuilds the local latest sequential order of committed transactions by running the algorithm update_sequential_order(), and updates the read set, write set and timestamps of both sets of committed transactions.
- Once a local or sub-transaction commits, it tries to remove committed transactions (or called outgoing nodes in the local serialization graph), which are not serialized after any active/committed transaction, from the local sequential order of committed transactions.

4.3.2 The cluster head functionality

A cluster head has following tasks to accomplish:

- It receives global transactions from clients and divides them into sub-transactions, which are sent to appropriate participating servers based on the global schema.
- It runs 2PC to request the status of sub-transactions, and at the same time requests timestamps of the read set.
- Once it receives all successful messages of a global transaction, it begins to validate this transaction globally using SODA in a critical section. After validation, it sends the validation result to each participating server, and sends the final results to clients.
- When a global transaction commits, it updates its sequential order by running the algorithm update_sequential_order(), and adds the read set, write set and the timestamp of both sets.

- Once a global transaction commits, it tries to remove committed transactions (or outgoing nodes in the global serialization graph), which are not serialized after any active/committed transaction, from the sequential order of committed global transactions.
- It periodically checks its remaining power level. If the level is below a predefined threshold, it resigns its cluster head role and elects a new cluster head in the neighborhood, so that the power consumption among servers is balanced.

5. PERFORMANCE EVALUATION

The simulation experiments are conducted to compare the performance of our proposed SODA with those of SESAMO [3] and the most widely used CC protocol - S2PL (Strict 2-Phase Locking). Here, we use the names of CC techniques to represent the names of transaction management techniques because CC is the most vital component in a transaction management technique. As we discussed earlier, SESAMO relaxes the atomicity and global serializability due to its assumption. The global serializability is guaranteed by S2PL when S2PL is combined with 2PC.

Our simulation model consists of a transaction generator, a real-time scheduler that schedules transactions using early deadline first, participating servers, coordinating servers or cluster heads for SODA only, and a deadlock manager for SESAMO and S2PL. In the SODA model, a transaction T issued by a client is distributed to its cluster head; the cluster head divides T into several sub-transactions, and transmits them to the appropriate participating servers according to the global schema. Each participating server processes the sub-transactions locally, and sends the results back to the cluster head. The cluster head runs the 2PC, gathers all results from the participating servers and validates T globally.

The simulation models for SESAMO and S2PL are similar to that SODA except for a couple of points. One is that SODA is applied locally and globally to validate transactions, while in SESAMO, strict 2PL is applied globally [3] and locally [11], and in S2PL, strict 2PL is only run only locally. The other point is that SESAMO and S2PL use coordinating servers instead of cluster heads.

5.1 Performance Metrics and Simulation Parameters

Our simulation is implemented using the AweSim simulation language [Pritsker, 1999]. Global transactions are defined as entities, and mobile hosts are defined as resources with different initial energy levels and randomly distributed locations. The simulation parameters are shown in Table 1. Due to space limitation only three performance metrics are presented in this paper: abort rate defined in Formula (5.1), total energy consumed by all servers defined in Formula (5.2) and average difference in energy consumption between two servers defined in Formula (5.3). Since in mission-critical applications, transactions should be executed not only correctly but also within their deadlines, that is, we use real-time transactions to evaluate the performance. Therefore, in our simulation, a transaction will be aborted if either it misses its deadline or the system could not complete it successfully (e.g. when it is aborted by the CC technique).

$$Abort\ rate = \frac{Total\ number\ of\ aborted\ transactions}{Total\ number\ of\ generated\ transactions} * 100\% \quad (5.1)$$

The second performance metric is the total amount of energy consumed by all servers. This metric evaluates how the energy efficiency of each technique is, and is computed with the following formula, where n is the total number of servers, and x_i is the energy consumed by server S_i.

$$Total\ energy\ consumed\ by\ servers = \sum_{i=1}^{n} x_i \quad (5.2)$$

The third performance metric is the average difference in energy consumption between two servers to evaluate how balanced the system is in terms of energy consumption. The more balanced the system is, the longer lifetime the system has. The average difference in energy consumption between two servers is computed in the following formula, where n is the total number of servers, and x_i *and* x_j are total energy consumed by servers S_i and S_j, respectively.

$$Average\ difference\ in\ energy\ consumption = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} | x_i - x_j |}{(n-1) * n} \quad (5.3)$$

Table 1. Simulation Parameters

Parameters	Default Value	Range
Simulation Area	1000x1000	-
Number of Clients	40	-
Number of Servers	20	-
Arrival Rate (transactions/second)	50	10 ~ 100 (Exponentially distributed)
Disconnection Probability	0.5	0.1 ~ 1.0
Read-only Transaction Probability	0.8	0.5 ~ 1
Percentage off Disconnection Probability for Cluster Heads only	50%	10% ~ 100%
Total transactions	1000	-

5.2 Simulation Results

Figure 3 shows the abort rates of SODA, SESAMO and S2PL increase when the transaction arrival rate increases. The abort rate of SODA is much lower than those of SESAMO and S2PL mainly because SODA is optimistic and non-blocking, and SESAMO's abort rate is slightly lower than S2PL's. Although SESAMO does not enforce global serializability, strict 2PL running both locally and globally still blocks many conflicting transactions. S2PL runs strict 2PL locally only, but it enforces global serializability using 2PC, so that all locks of sub-transactions are held until global transactions commit, and this situation increases the waiting time of conflicting transactions in S2PL. When the arrival rate is getting larger, abort rates of

SESAMO and S2PL are almost the same. Also it is easy to see that the abort rates of SODA are getting worse quickly when the contention of transactions increases.

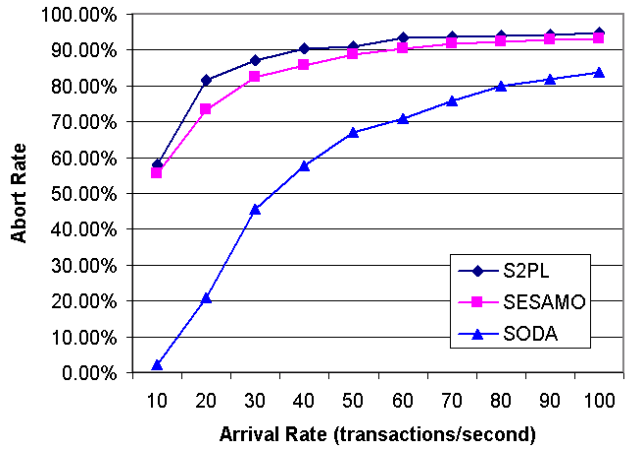

Figure 3. Abort rate vs. arrival rate

Figure 4 shows the abort rates of the three techniques increase as the disconnection probability increases. Disconnection probability 0.5 means that 50% of communications become disconnected when a mobile node tries to communicate with other nodes. One of the major characteristics of MANET is frequent disconnections due to the mobility and energy limitation of nodes, and unreliable wireless communication between nodes. Frequent disconnections of a mobile node may result in long-lived transactions and may decrease the performance of MANET databases. Therefore, it is very important to show the proposed SODA still works more efficiently even though when the disconnection probability is high.

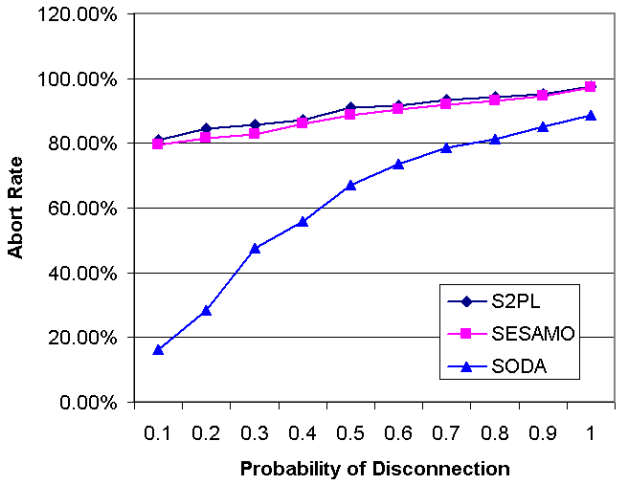

Figure 4. Abrot rate vs. probability of disconnection

As shown in Figure 4, the abort rate of SODA is at least 8% lower than those of SESMO and S2PL because SODA uses stable cluster heads as coordinating servers, is optimistic so that it does not block transactions, and utilizes the dynamic adjustment of sequential order of committed transactions to reduce aborts. The abort rate of SESAMO is still slightly lower than S2PL's due to the same reason discussed above, and they are almost the same when the probability is 1. Since SESAMO has at least 79% abort rate, this shows that SESAMO does not consider frequent disconnections.

Figure 5 shows that the total energy consumption of all servers increases with the increase of the arrival rate. This is trivial because each server has to work hard to process transactions when more transactions are initiated. When the arrival rate is greater than or equal to 20, SODA consumes at least 15 W less than both S2PL and SESAMO and at the same time SODA has less abort rate as shown in Figure 3. This contributes to SODA is optimistic and does not hold limited system resource to prevent conflicting transactions access them, and also stable cluster heads work as coordinating servers.

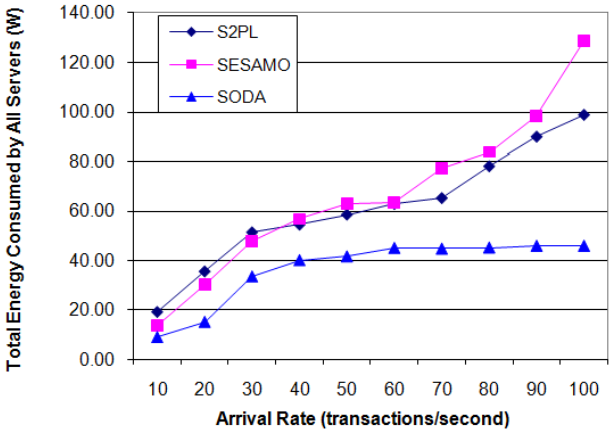

Figure 5. The total energy consumped by servers vs. arrival rate

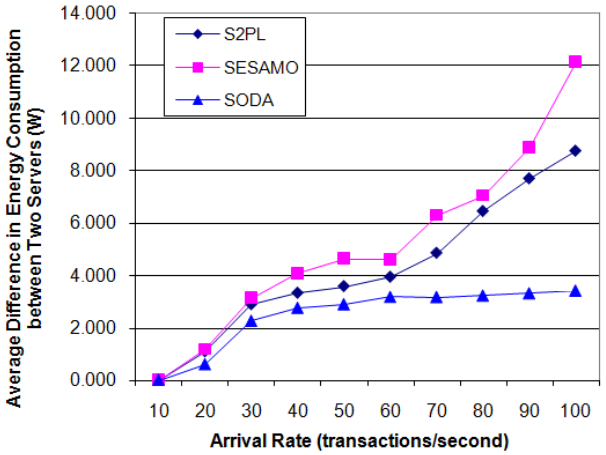

Figure 6. The average difference in energy consumption between two servers vs. arrival rate

The average difference in energy consumption between two servers by varying the arrival rate is shown in Figure 6. Through this metric, we want to check if the energy consumption is balanced among servers. If a technique does not balance energy consumption among servers, some server may run out of power quickly and consequently affects the whole database system. It is

easy to see that SODA balances the energy consumption best because clients are grouped into clusters and always submit their transactions to their own cluster heads; however, in SESAMO and S2PL, they submit transactions randomly so that some servers are overloaded.

6. CONCLUSTIONS AND FUTURE WORK

In this paper, we introduced a transaction management technique that can be used to support mission-critical applications such as disaster rescue and battlefields. This technique considers transaction real-time constraints as well as mobility and energy limitation of both servers and clients. Our solution is aimed at reducing the transaction abort rate while saving the energy consumption by servers and balancing the energy consumption among servers, and the simulation results indeed confirm that we reach our goal.

For future research, we plan to incorporate data replication into our model to improve the data access time and availability. We will also investigate the effects of mobility (speed) of mobile nodes and number of servers on these three performance metrics.

7. REFERENCES

[1] Alampalayam, S. P., and Srinivasan, S., 2009. " Intrusion Recovery Framework for Tactical Mobile Ad hoc Networks", the International Journal of Computer Science and Network Security, VOL.9 No.9, September 2009.

[2] Basu, P., Khan, N., and Little, T. D. C., 2001. “A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks,” In Proceeding of IEEE ICDC, April, 2001, pp. 413-418.

[3] Brayner, A., and Alencar, F. S., 2005. “A Semantic-serializability Based Fully-Distributed Concurrency Control Mechanism for Mobile Multi-database Systems”, Proceeding of the 16th International Workshop on Database and Expert Systems Applications (DEXA), 2005, pp. 1085-1089.

[4] Catarci, T., de Leoni, M., Marrella, A., Mecella, M., Salvatore, B., Vetere, G., Dustdar, S., Juszczyk, L., Manzoor, A., and Truong, H. 2008. “Pervasive Software Environments for Supporting Disaster Responses,” IEEE Internet Computing, Vol. 12, No. 1, 2008, pp.26-37.

[5] Chatterjee, M., Das, S. K., and Turgut,, D., 2002. “WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks,” Cluster Computing, Vol. 5, 2002, pp. 193-204.

[6] Chlamtac, I., Conti and, M., Liu, J., 2003. “Mobile Ad Hoc Networking: Imperatives and challenges”, Ad Hoc Networks Publication, Vol. 1, Issue 1, July 2003, pp. 13-64.

[7] Dirckze, R., and Gruenwald, L., 2000. “A pre-serialization transaction management technique for mobile multi-databases,” Special Issue on Software Architecture for Mobile Applications, MONET 2000, pp. 311–321.

[8] Fei, Y., Zhong, L., and Jha, N. K., 2008. “An energy-aware framework for dynamic software management in mobile computing systems,” ACM Trans. on Embedded Computing Systems, April 2008

[9] Fudickar, S. J. F., and Rebensburg, K., 2007. “Mobile Group-based Multimedia Cooperation System for MANETS,” IEEE Multidisciplinary Engineering Education Magazine, June 2007, Vol. 2, No. 2, pp. 51-54.

[10] Gruenwald, L., Banik, S. M., and Lau, C., 2007. "Managing real-time database transactions in mobile ad-hoc networks", Distributed and Parallel Databases journal, Vol. 22, No. 1, August 2007.

[11] Holanda, M., Brayner, A., and Fialho, S., 2008. “Introducing self-adaptability into transaction processing”, SAC 2008, pp. 992-997.

[12] Lu, W., Seah, W. K. G., Peh, E. W. C., and Ge. Y., 2007. “Communications Support for Disaster Recovery Operations using Hybrid Mobile Ad-Hoc Networks,” Proceedings of the 32nd IEEE Conference on Local Computer Networks, Dublin, Ireland, 2007, pp. 763-770.

[13] Obermeier, S., Böttcher, S., Hett, M., Chrysanthis, P. K., and Samaras. G., 2009. Blocking reduction for distributed transaction processing within MANETs. Distributed and Parallel Databases 25(3): 165-192 (2009).

[14] Silberschatz, A., Korth, H. F., and Sudarshan, S., 2005. Database Systems Concepts, McGraw-Hill College.

[15] Sklavos, N., Touliou, K., 2007. "Power Consumption in Wireless Networks: Techniques & Optimizations", proceedings of The IEEE Region 8, EUROCON 2007, International Conference on "Computer as a Tool" (IEEE EUROCON'07), September 9-12, 2007.

[16] Viswacheda, D. V. , Arifianto, M. S., and Barukang, L., 2007. “Architectural Infrastructural Issues of Mobile Ad hoc Network Communications for Mobile Telemedicine System,” In Proceeding of 4th International Conference on Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia, 2007.

[17] Xing, Z., Gruenwald, L., and Phang, K. K., 2008. "SODA: an Algorithm to Guarantee Correctness of Concurrent Transaction Execution in Mobile P2P Databases", Proceedings of the 19th International Conference on Database and Expert Systems Application Workshop, September 2008, pp. 337-341.

[18] Xing, Z., Gruenwald, L., and Phang, K. K., 2010. “A Robust Clustering Algorithm for Mobile Ad-hoc Networks”. To appear in the book “Handbook of Research on Next Generation Networks and Ubiquitous Computing”, IGI Global, 2010, in press.