A Caching Model for Real-Time Databases in Mobile Ad-Hoc Networks

Yanhong Li and Le Gruenwald*

School of Computer Science, University of Oklahoma
Norman, OK 73072, USA
{yanhong.li-1, ggruenwald}@ou.edu
http://www.cs.ou.edu/~database

Abstract. Although caching has been shown to be an efficient technique to improve the performance of database systems, it also introduces the overhead and complexity in maintaining data consistency between the primary copies on servers and the cached copies on clients. Little research has been performed for data caching in the mobile ad-hoc network (MANET) environment where both servers and clients are nomadic. In this paper, a caching model called GMANET is designed to maintain both strong and weak cache consistency for distributed real-time database transaction systems in group-based MANETs, and at the same time, to incur as few update control messages as possible. GMANET is compared with the existing caching models by means of simulation. The experiment results show that the GMANET has the best performance in terms of percentage of transactions processed before their deadlines and is compatible with other caching models in terms of mobile hosts' energy consumption.

1 Introduction

With the advances in wireless networking technology and portable mobile devices, a new computing architecture called mobile ad hoc wireless networks (MANETs) is emerging. Applications in MANET are typically those that require the rapid deployment of mobile hosts and occur in a situation where a fixed infrastructure is not available. Example applications include military operations and disaster relief efforts.

Mobile hosts in MANETs are powered by short-lived batteries, communicate via an unreliable wireless link, and move in various speeds. As a result, these mobile hosts may experience severe network congestion, prolonged transaction execution, or even frequent abortion of the transactions. These additional restrictions plus the deadline constraints imposed on time-critical applications call for a new power-aware and communication-cost efficient caching technique for real-time MANET database system. Developing such a technique is the objective of our research.

Caching has been proven to be an essential technique for improving the performance of many computing environments, such as network file systems, wired distributed database systems, and web applications [3]. The purpose of caching is to bring the data source as close to clients as possible, and thus, save a round-trip when the re-

* This work was partially supported by the National Science Foundation grant No. IIS-0312746.

K.V. Andersen, J. Debenham, and R. Wagner (Eds.): DEXA 2005, LNCS 3588, pp. 186 – 196, 2005.

quested data are found in the local cache storage [1]. However, maintaining cache consistency is a challenging problem. Cache consistency can be categorized into two types: tight/strong cache consistency and loose/weak cache consistency [3, 14]. Strong cache consistency refers to the caching techniques that can always maintain consistency between the cached data and the original ones. Weak cache consistency refers to those that allow the data divergence between the cached data and the original ones.

The rest of this paper is organized as follows. Section 2 reviews the current caching techniques in mobile database systems. In Section 3, GMANET, a caching model for group-based MANETs, is proposed. Section 4 reports simulation results. Section 5 presents conclusions and future research work.

2 Literature Review

Several proposals have been made to solve the cache consistency problem in mobile databases. The invalidation report technique in which the servers broadcast the invalidation reports to their clients periodically was proposed to maintain strong cache consistency [2,7]. But, it incurs the query latency [2,7] and tremendous communication cost. The refresh time strategy aimed at maintaining weak cache consistency was proposed in [4]. However, it does not guarantee the freshness of the cached data so database transactions may access the dirty data.

Cooperative caching in MANET, which allows a client to access the cached data of its neighbors, was proposed in [13]. The Time-To-Live (TTL) mechanism in [13] is used to maintain the weak consistency level of the cache. Again, it is not applicable for transactions that need accurate data. Another caching model, called MANET caching, was proposed in [10]. The refresh time strategy was adopted from [4] and modified to maintain the weak cache consistency. However, the cached data can be only used for the read-only transactions that can tolerate out-dated data.

In reality, the mobile clients in many applications, such as battlefields, medical emergencies, and fire-fighting operations, are organized in groups and their movements follow pre-defined patterns [6] instead of total randomness of mobility like the mobility model used in [10]. Thus, a new caching model called GMANET for a group-based MANET is proposed in this paper and also takes power consumption, bandwidth, and real-time constraints into consideration.

3 The GMANET Caching Model

3.1 The GMANET Architecture

The GMANET architecture is illustrated in Fig. 1. Similar to the environment in the MANET caching model in [10], the proposed group-based MANET also consists of two representative devices, Large Mobile Hosts (LMHs) such as laptops, and Small Mobile Hosts (SMHs) such as PDAs. The group-based MANET has a number of groups. Each group logically has the following entities: group leader LMH (LMHg), ordinary LMH and group member SMH. The LMHgs and LMHs have the whole database management system and SMHs has a caching and query processing module.

The Location-Aided Routing protocol (LAR) [8] is assumed to carry out routing packets from the sending MH to the receiving MH. The groups, group leaders, and group members are defined by applications. If a group leader fails or needs to recharge its power, it will designate another LMH in its group as a deputy group leader until it recovers and then assumes the group leader role again.

Hereafter in this paper, clients refer to those SMHs that initialize transactions and send to servers, and servers refer to those LMHs that provide data service to other network members. When LMHs request data from other servers, they themselves become clients.

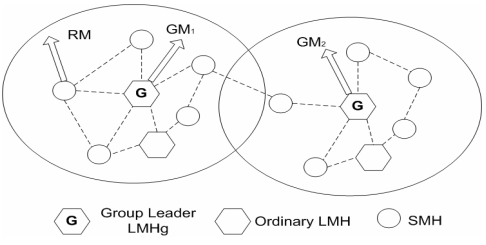


Fig. 1. The GMANET architecture modified from [6]

3.2 Data Access and Update Model

The data model proposed in [10] is adapted in this paper. According to the update characteristics, data are classified into the Periodically Updated data (PU) and Randomly Updated data (RU). The periodically updated data are the data that get updated at fixed update intervals that are specified by applications. Obviously, the periodically updated data are valid to use before their next updates. In battlefield applications, the information about logistics of all battalions is an example of a periodically updated data (PU). In contrast, the randomly updated data (RU) are updated anytime. An example of such data is the current number of refugees in the emergency and rescue operations. The purpose of this data update model is to support the different data freshness requirements of different transactions.

3.3 Transaction Type

In our real-time database system, transactions can be either firm or soft, which is defined by database applications. Firm transactions are aborted if they missed their deadlines while soft transactions continue execution unless they missed their second deadlines. Each transaction consists of a set of read and/or write operations. It has been observed that most of the applications in mobile environments generate more read operations than write operations [10]. All the write transactions are assumed to be executed on the servers; while the read-only transactions can be executed by accessing the cached data items. In some applications, some read transactions might not be as critical as the other read transactions with regard to data freshness. For example, from a driver's viewpoint, the weather information and the traffic information measured at noon is not much different than those measured at 12:05 PM. But, the information about the location, emergency medical care, or accident investigation should

be as accurate as possible [9]. Therefore, the read-only transactions in [10] are further divided into two types: 1) Up-to-Date (UD) type read-only transactions that accept only fresh data (e.g. transactions querying data about locations of enemies) and 2) OD (Out-Dated) type read-only transactions that accept slightly stale data (e.g. transactions requesting data about logistics of battalions).

3.4 The Caching Process in GMANET

3.4.1 The Cache Consistency

In the MANET caching model [10], the caches on the servers and clients are both maintained at the weak consistency level by using the refresh time strategy, thus preventing all the UD type read transactions from using the cached copies. As a result, The UD type read transactions have to be sent and distributed to the original servers, lengthening the processing of these transactions. Therefore, the efficiency of the MANET caching model depends largely on the application requirements. I

In our proposed GMANET caching model, both the strong and weak cache consistency levels will be maintained. The cache on clients will be maintained at the weak consistency level by the refresh time strategy and the cache on group leaders will be maintained at the strong consistency level by an asynchronous invalidation strategy. GMANET with both cache types solves the above shortcoming of the MANET caching model. These two types of caching mechanism are discussed separately as follows.

3.4.1.1 Weak Cache Consistency on Clients

Clients are allowed to cache the previously accessed data items so that the subsequent requests may be satisfied by the cached data and thus avoid sending them to servers. The refresh time strategy in [4] is modified to keep the cached data consistent at the weak consistency level on the client side (SMHs and ordinary LMHs) in the GMANET caching model. Each cached data item is associated with a refresh time indicating how long this particular data item remains valid in the client's cache. In order to calculate the refresh time, the update log containing the statistics about the update pattern is maintained on each data server. The update log records the data id, the previous mean refresh time for this data item, and the latest update timestamp of all data residing on the server. Before servers return the transaction results to clients, they estimate a refresh time for each data item in the transaction result and the estimated refresh time will be sent along with the transaction result back to clients.

The refresh time calculation is modified as follows to fit the GMANET caching model: 1) $FRT_i = T_i + \overline{d_i} - T_{comm}$ for randomly updated data, 2) $PRT_i = T_i + P_i - T_{comm}$ for periodically updated data, 3) T_{comm} = *NoOfHops* * *PackageSize* / *Bandwidth*, where T_i is the current update timestamp of data item i, $\overline{d_i}$ is the mean update duration on data item i, P_i is the fixed update interval, T_{comm} is the transmission time between the sender and receiver, *NoOfHops* is the number of hops between the sender and the receiver, *PackageSize* is the transmission amount, and Bandwidth is the wireless bandwidth. The reason that the refresh time is sub-

tracted by the communication cost is to reduce the effect of the communication time, and thus, reduce the staleness degree of the cached data when transmitting from servers to clients.

Through the refresh time strategy the clients are not relying on the servers' help to validate their cached data items because each cached data item has already been attached with the refresh time specifying how long it is valid in the future. It means that no additional communications between the clients and servers are needed. Unlike the invalidation strategy where the clients have to be connected and tuned in to receive the invalidation reports from servers periodically or asynchronously, this method allows clients to be free to move and disconnect (offline) and still be able to validate the cached data items when the clients access their cached data. However, the refresh time strategy cannot maintain strong cache consistency while the invalidation strategy does.

3.4.1.2 Strong Cache Consistency on Group Leaders

Group leaders are allowed to cache the passing-by data on behalf of their clients. This is because all the transactions initiated from clients are first sent to group leaders and, therefore, group leaders can see all the network traffic within their registered clients. Thus, LMHgs are selected as the locations where the cached data is maintained at the strong consistency level and valid for access anytime. This will improve the processing of UD type read-only transactions.

The caches on group leaders are kept fresh by relying on the combination of invalidation and refresh time techniques. From the review on strong cache consistency in Section 2, maintaining the cache at the strong consistency level is quite expensive since it requires all the updates made on the servers be propagated to the cache holders immediately. In GMANET, of all the LMHs we assume only group leaders are allowed to maintain their cache at the strong consistency level for their clients, thus cutting down the total number of sites that need invalidation messages. As a result, the communication overhead to maintain strong cache consistency is tolerable since the number of group leaders is much smaller than the number of LMHs in GMANET.

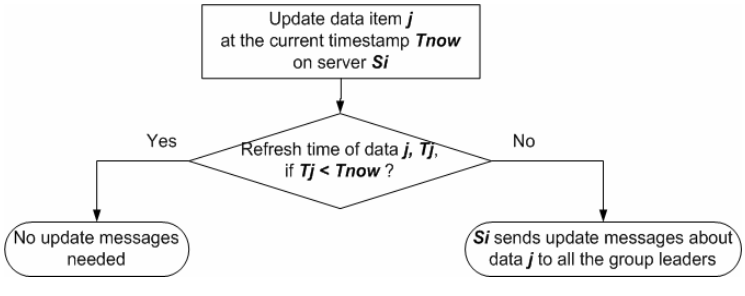


Fig. 2. The invalidation messages from LMHs to LMHgs

The communication overhead to maintain strong cache consistency by invalidation technique can be further reduced by making use of the refresh time for each cached data item. As discussed in the previous section, each data server maintains an update log, which is used to estimate refresh time for each accessed data item for its clients. The data servers also use the refresh time to reduce the update or invalidation mes-

sage exchanges between them and the group leaders. As shown in Fig. 2, when data is updated on an data server, it first looks up its update log to see whether the refresh time of this updated data item is expired or not. If expired, then it is not necessary to send the update message to group leaders; if not expired, then the data server must send the update messages to each of the group leaders that have cached the updated data item. For example, if 50% of the cached data are associated with the correct refresh time, then the update messages will be cut down by 50%, compared with the cost of traditional invalidation methods to maintain strong cache consistency, where every update invokes a communication for propagating the updated value to all the cache holders.

It can also be seen that the tighter the estimated refresh time, the fewer update messages necessary to keep the cache consistent with the original servers. The tight estimation of the refresh time means that the refresh time tends to be small and expired before the actual update, and thus the data servers do not need to send out update messages, saving a lot of bandwidth and energy to transmit these control messages.

When a network partition occurs, the delayed update is assumed by the data servers. It means the data servers will wait to receive all acknowledgements of the group leaders before the actual updates are committed. If some group leaders are in a different partition, the data servers will delay the updates until the refresh time expires.

3.4.2 Cache Data by Clients and by Group Leaders

When a transaction result is returned to a client from its group leader, the client will check the transaction type since the client only caches the data requested by OD type read transactions; the data items accessed by UD type read transactions have already been cached by their group leaders. The reasons for this cache assignment are two folds. First, it is to reduce the cache storage burden on the client side since clients are much more limited in terms of memory and disk space. Second, it is to reduce the redundancy since it is not efficient if both clients and their group leaders cache the same data. For each data item marked with the refresh time in the transaction result, if it is already cached, the client updates its value and refresh time; if it is not cached and the cache storage is not full, the client inserts it into an empty space; otherwise, the client executes the replacement policy (Section 3.4.5) to select a slot for a newly arrived data item.

The group leaders will cache only data that are accessed by UD read-only transactions. When the results of all the sub transactions are sent back from all the participating servers (LMHps), the group leaders have a decision to make as to what kind of data they should cache. In order to reduce the overhead of the communication cost associated with the strong cache consistency policy, the group leaders only cache the data accessed by UD type read-only transactions instead of all the transactions. The accessed data by the OD type read-only transactions will be cached on the clients' side as discussed above. The cache on the group leaders will be used to satisfy all the subsequent UD read-only transactions sent from their group members.

3.4.3 Cache Access by Clients

In GMANET, the cache on the client side is only kept at the weak consistency level, and thus only OD type read transactions accepting slightly out-dated data can access it

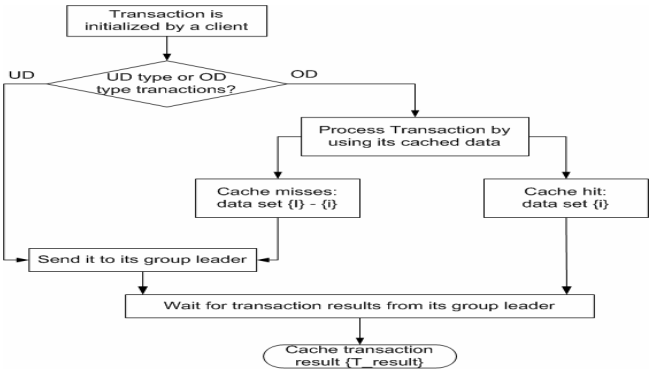


Fig. 3. The access of cached data items on the client side

(see Fig. 3); the cache on the group leaders is kept at the strong consistency level, and thus, read transactions, either OD or UD type, can access the cached data on LMHgs. All the write transactions have to be sent and processed at the group leaders LMHgs.

It is highly likely that the OD type read transactions are answered by using locally outdated cached data because the weak consistency protocol is used on the client's side. Even though the servers help estimate the refresh time based on the past update statistics of the clients, the refresh time of each cached data item is only the best effort estimation. Any updates happened on the servers may not reflect on the cached copies of the clients in a timely fashion, and thus, it is inevitable for OD type read transactions to access the stale cache.

The access of the cached data items by clients is shown in Fig. 3. If an initiated transaction is an OD type read transaction, the client checks its cache first and identifies all the cache misses and cache hits. It then sends the transaction requesting for cache misses to the group leader. If it is the UD type read-only or write transaction, the client sends it to its group leader for processing directly.

3.4.4 Cache Access by Group Leaders

When a transaction is initialized by an SMH and it is an OD read-only transaction, it is first processed by the SMH to identify the locally qualified cached data items called cache hit or cache miss otherwise. After processing, the SMH will package the cache misses and send them to its group leader. Since the cached data on LMHgs are always kept consistent with the original copies, all the read transactions, both UD and OD types, can access the cached data.

The cache access by group leaders is shown in Fig. 4. When a new transaction arrives, and it is the read transaction, it will be first processed by using the cached data maintained on the group leaders, and the cache hit data set, which is {i'}, and cache miss data set, which is {I} – {i} – {i'} (note the {i} is the local cache hit data set on the transaction initiator client), are identified. The group leader then distributes the cache misses to all the participating servers (LMHps). All the write transactions are distributed to the original servers for processing.

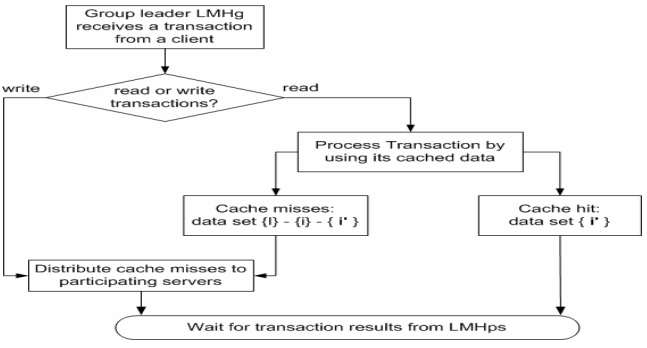


Fig. 4. The access of cached data on the group leader side

3.4.5 Cache Replacement

The cache replacement proposed in [10] is adapted in this research to make room for newly arrived data items when the cache storage is full. Both the group leaders and their group members use the same replacement policy. Initially, each mobile host (server or client) will check whether its storage is full or not. If it is not full, it inserts the newly arrived data into the empty entries in its cache. If it is full, it starts the replacement process by searching for expired data items in its cache. If an expired item is found, it just replaces it; otherwise it searches for the data items with the lowest access frequency accessed by soft transactions. If it is not found, then it searches for the data items with the lowest access frequency by firm transactions. The unique characteristic of this replacement policy is that it puts a higher caching priority on firm transactions than soft transactions, and thus, favors firm transactions over soft transactions since the firm transactions must be aborted if they missed their deadlines.

4 Simulation Experiments

Four simulation models using AweSim software [11] are built to compare the GMANET caching model, the MANET caching model, CHAN caching model and the baseline model without caching module. The performance metrics are 1) the percentage of transactions missing their deadlines, indicating how many transactions cannot be processed successfully within their transaction deadline requirements [10], 2) the total energy consumptions of all LMHs and all SMHs, 3) the average difference in energy consumption between two LMHs indicating how balance the system is in terms of energy consumption, and 4) the cache hit ratio, computed as the transactions fulfilled by the servers' or clients' caches over the total transactions initialized in the system. The parameters listed in Table 1 are the default parameter settings for the following simulation experiments.

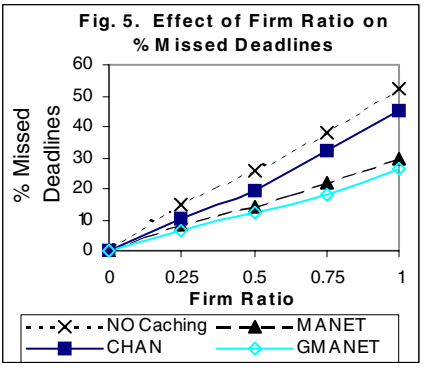
4.1 Effect of Firm/Soft Ratio

The four models show different capacities of processing transactions before their deadlines in Fig. 5. The NO caching model performs the worst, followed by the

Table 1. The simulation parameters in the system

Parameter	Value Range	Default	Reference
Firm/Soft Ratio	0, 0.25, 0.5, 0.75, 1	0.5	
OD/UD Ratio	0, 0.25, 0.5, 0.75, 1	0.75	
Cache Size	0.2, 0.4, 0.6, 0.8	0.6	
Number of LMHs	-	20	
Number of SMHs	-	40	
Simulation Area	-	1000x1000	
Bandwidth	-	11 Mpbs	[12]
LMH Energy Dissipation Rate in Active Mode	-	15.4 w	[5,12]
LMH Energy Dissipation Rate in Doze Mode	-	7.97w	[5,12]
SMH Energy Dissipation Rate in Active Mode	-	2.178 w	[5,12]
LMH Energy Dissipation Rate in Doze Mode	-	1.4w	[5,12]

CHAN caching model and the MANET caching model; the best is the GMANET caching model. With more firm transactions in the system, the performance gaps among these four models become larger. It is obvious that the NO caching model is the worst because it does not equip with the caching component like the other three caching models. The better performance of the three caching models is contributed to the caching mechanism so that some transactions can reduce their transaction paths by going through shortcuts of the cached copies instead of the whole transaction path. Also, the shortened transaction path means a reduced amount of transmission, thus saving a lot of energy spent on transmission and computation. Among the three caching models, the GMANET has the best performance due to the fact that its caching system allows not only OD type read transactions but also UD type read transaction to access its cached data. Compared to GMANET, the MANET and CHAN Caching models allow only OD type read-only transactions to access their cached data, and require all the UD type read-only transactions to be processed by the original servers.

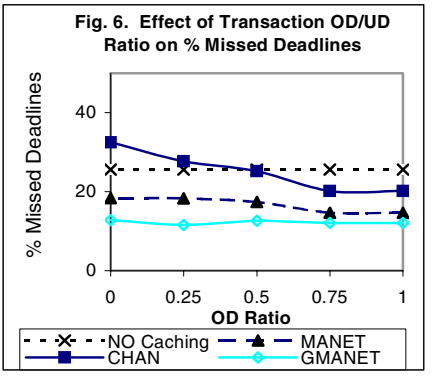


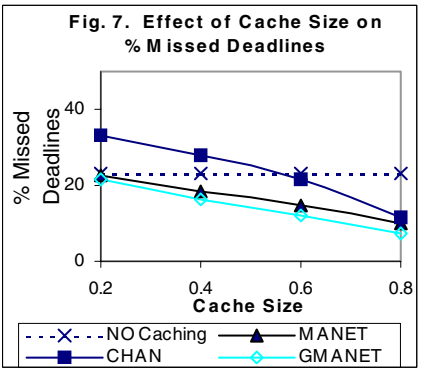
4.2 Effect of OD Ratio

Fig. 6 shows that when the OD type transaction increases, the percentage of transactions missing deadlines decreases in both CHAN and MANET, but it remains more or less the same in the GMANET caching model, and does not exhibit as much sensitivity to the changes of OD type transactions as the other two caching models do. The above observed trend is expected. As all the caches in both the CHAN and MANET caching models are maintained at the weak consistent level, only OD type transactions are allowed to access the cached data and all the UD type transactions will be sent to the origin server to process. While the cache on the group leaders in GMANET is maintained at the strong consistency level and the cache on the clients is maintained at the weak consistency level, the OD type transactions can access the cached data on the clients and on the group leaders, and the UD type transaction can access the cached date on the group leaders. Thus, the performance gain in GMANET is contributed to its double cache types in the system. The performance in terms of percentage of transactions missing deadlines of GMANET is on average about 20% better than MANET, 50% better than CHAN, and 100% better than NO Caching.

4.3 Effect of Cache Size

The cache size on the clients and servers is varied to show its effect on the performance of the four models. The NO caching model does not change with the cache size in all the metrics studied because it has no caching mechanism. It can be seen in Fig. 7 that the percentage of the transactions missing deadlines decreases with the increase of the cache size in all the three caching models. With the increase of the cache size more data can be cached for the subsequent transactions, and thus, the probability of satisfying the transactions with the local cached data increases. Among the three caching models, the GMANET technique performs the best, followed by MANET and CHAN. The same analysis as that in the previous experiments holds true for the performance differences among the three caching models.





5 Conclusions

Designing a caching technique in group-based MANETs is meaningful since, in practice, with most applications such as battlefields, medical emergencies, and fire-fighting, there are several logical units involved and their movements follow some pre-defined patterns instead of total randomness. A new caching model called GMANET has been proposed for group-based MANETs in this research, and the preliminary simulation results show that it performs the best in terms of its ability to complete transactions before their deadlines. However, a hand-off mechanism is needed to handle the scenario when the group leaders run out of energy and new group leaders are designated as their successors.

References

1. Baentsch, M., Baum, L., Molter, G., Rothkugel, S., and Sturm, P.: Enhancing the Web's infrastructure: from caching to replication. *IEEE Internet Computing*. 1 (1997) 18-27
2. Cao, G.: On Improving the Performance of Cache Invalidation in Mobile Environments. *Mobile Networks and Applications*, 7(2002) 291-303
3. Cao, L.Y., and OZSU, M.T.: Evaluation of Strong Consistency Web Caching Techniques. *World Wide Web: Internet and Web Information Systems*, 5 (2002) 95-123
4. Chan, B-Y., Si, A. and Leong, H-V.: A Framework for Cache management for Mobile Databases: Design and Evaluation. *Distributed and Parallel Databases*, 10 (2001) 23-57
5. Feeney, L. M., Nilsson, M.: Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *Proc. of IEEE INFOCOM.* Anchorage, Alaska, (April 2001) 1548-1557
6. Hong, X., Gerla, M., Pei, G., Chiang, C-C.: A Group Mobility Model for Ad Hoc Wireless Networks. In *Proc. of the 2nd ACM int'l workshop on Modeling, analysis and simulation of wireless and mobile systems.* Seattle, Washington (August 1999) 53-60
7. Jing, J., Elmagarmid, A., Helal, A. and Alonso, R.: Bit-sequences: An Adaptive cache invalidation method in mobile client/server environments. *Mobile Networks and Applications*, 2 (1997) 115-127
8. Ko, Y-B., Vaidya, N. H.: Location-Aided Routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6 (2000) 307-321
9. Lam, K-y., Chan, E., Yuen, Joe C-H.: Broadcast Strategies to Maintain Cached Data for Mobile Computing System. In *Proc. of the Workshops on Data Warehousing and Data Mining: Advances in Database Technologies*. Springer-Verlag, London, UK(1998) 193-204
10. Lau, C.: Handling Mobile Host Disconnection, Data Caching, and Data Replication in Managing Real-Time Transactions for Mobile Ad-Hoc Network (MANET) Databases. *Master Thesis, University of Oklahoma*, Norman, OK (2002)
11. Prisker, A. Alan B., O'Reilly, Jean J.: Simulation with Visual SLAM and AweSim. *Wiley System Publishing Corporation,* West Lafayette, Indiana (1999)
12. Shih, E., Bahl, P., Sinclair, M.J.: Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Proc. of the Eighth Annual Int'l Conf. on Mobile Computing and Networking*. Atlanta, Georgia (September 2002) 160-171
13. Yin, L. and Cao, G.: Supporting Cooperative Caching in Ad Hoc Networks. In *Proc. of the 5th ACM int'l workshop on Wireless Mobile Multimedia*. Atlanta, Georgia(Sep.2002) 56-63
14. Yu, H., Breslau, L., Shenker, S.: A Scalable Web Cache Consistency Architecture. In *ACM SIGCOMM Computer Comm. Rev., Proc. of the conf. on Applications, technologies, architectures, and protocols for computer communication.* Cambridge, MA (1999) 163-174