Evaluation of a Criticality-Based Method for Generating Location Updates[1]

Özgür Ulusoy*, İlker Yoncacı*, Kam-yiu Lam†

*Department of Computer Engineering, Bilkent University, Ankara, Turkey.
†Department of Computer Science, City University of Hong Kong, Hong Kong.
E-mail: {oulusoy, yoncaci}@cs.bilkent.edu.tr, cskylam@cityu.edu.hk

Abstract: In a mobile computing environment, the result of a *location-dependent query* is determined by the current location of the user who has issued the query, as well as the locations of moving objects on which the query has been issued. If the location-dependent query is submitted as a *continuous query*, the result of the query changes as the user and/or the queried moving objects move. The result of a *location-dependent continuous query* (LDCQ) can be provided to the user as a set of tuples $< S, begin, end >$ indicating that object S satisfies the query from time $begin$ to time end. The accuracy and timeliness of results of LDCQs have been studied in a previous work by employing an *adaptive monitoring method* (AMM) that manages the locations of moving objects. In this paper, we propose a *categorized adaptive monitoring method* (CAMM) which is based on AMM, and which allows users to specify various criticality levels for LDCQs. The aim of CAMM is to provide higher levels of accuracy for query results as the criticality of the query increases, without significantly increasing the wireless bandwidth requirements. We provide a simulation model with multiple criticality levels for LDCQs and evaluate the performance of the proposed method.

1 Introduction

A typical mobile database system can be characterized by a database server, a collection of moving objects and clients which can move while retaining their network connection through wireless links. Mobile clients represent users equipped with mobile units that can communicate and generate queries to be processed by the database server. When a user submits a *location-dependent query*, the answer to the query depends on both the location of the user and the locations of the moving objects on which the query has been issued [Si97, Wo97]. A location-dependent query can become more difficult to process when it is submitted as a *location-dependent continuous query* (LDCQ) for which the answer changes as the user and/or the queried objects move. Such a query is evaluated continuously by the database server for a specified period of time and the results changing dynamically can be transmitted to the requesting user.

Although there exist a considerable amount of work in the literature exploring LDCQs, none of those works investigates the issue of processing LDCQs that might be characterized by different priorities or criticalness. In our work, we deal with the processing of LDCQs based on a categorization of the queries according to their user-defined criticality. We believe that some of the queries in a mobile computing environment may need to produce more accurate answers than the others, and thus they are considered to be more

[1]This research is supported by the Research Council of Turkey (TÜBİTAK) under grant number 102E021.

$critical$. For example, the LDCQ "check for a free ambulance within 5 kilometers (to pick up a patient) of the hospital" should be more critical than the LDCQ "check for a free taxi within 10 kilometers (to pick a passenger)". In the former case, in order to cope with the emergency as soon as possible, the correct determination of ambulance locations is very crucial.

Producing accurate results for LDCQs is a difficult goal to achieve in a mobile environment, which requires an efficient management of location information. In this paper, we present a new method to monitor the locations of moving users/objects based on criticality of LDCQs so that higher levels of accuracy can be achieved for the results returned to the queries categorized with higher criticality. We also provide a detailed simulation model of a mobile computing system that supports processing of LDCQs associated with different criticality categories.

In the next section, we briefly discuss the background and related work. We describe in the same section a location update generation method, called Adaptive Monitoring Method (AMM) which our work is based on. We introduce the criticality-based Categorized Adaptive Monitoring Method (CAMM) in Section 3. We present the simulation model and performance evaluation results in Section 4. Concluding remarks are provided in the last section.

2 Background and Related Work

The problem of data dissemination using data broadcast and some possible solutions to it are studied in [AFZ96, Da97, FR98, IVB97]. The issues in cache invalidation and management are addressed in a number of articles such as [BJ96]. The problems associated with the indexing of dynamic attributes (such as location) in a mobile database system are addressed in [TUW98]. In that work, a variant of the quadtree structure for indexing dynamic attributes is proposed and an algorithm for generating the index periodically that minimizes the CPU and disk access cost is provided. In [KGT99], algorithms are proposed for indexing mobile objects in one and two dimensions using dynamic external memory data.

In [WL01], a stochastic model is provided to compute the optimal update boundary for the distance-based location update algorithm. Optimization issues in processing queries in mobile database systems are discussed in [KZ98]. In that work, the authors present a cost model for query optimization incorporating location information.

Issues related to location dependent query processing for moving objects have been addressed in a number of studies (e.g., [Si97, DK98, Wo99, RD00, SDK01]). In [Si97, Si98], a data model called $Moving\ Objects\ Spatio\text{-}Temporal$ (MOST) is introduced for databases containing position information about moving objects. MOST models the position of a moving object as a function of time. Therefore, the answer to the query: "retrieve the current position of the object S" in the MOST data model is different for time points t_1 and t_2 even if the value of the attribute specifying S's position has not been explicitly updated.

Consider the LDCQ "display motels within 5 miles of my position" issued by a person driving a car. When such a query is entered in the MOST data model, the query is evaluated once and a set of tuples is returned as the answer. The answer set consists of tuples $< S, begin, end >$ indicating that object S is the answer of the query from time $begin$ to time end. Once the answer to the query is computed, a decision has to be made in order to determine the time to send the tuples in the answer set to the user. In [GU00], we present a variety of approaches for the transmission of the tuples in the answer set of a LDCQ.

In [La01], we provide the $Adaptive\ Monitoring\ Method$ (AMM) for managing the locations of moving objects. AMM aims to provide high level of accuracy to the results of LDCQs. In this method, moving objects generate updates to report their current locations to the database server. The database server determines $update\ generation\ thresholds$ for the mobile clients that have submitted LDCQs and for the other moving objects in the system. Location of a mobile client is updated more frequently, if it has submitted a LDCQ and the query is still being processed.

In determining a location update for a moving object, AMM considers the deviation of the actual and computed location values of the object. If the deviation is greater than a certain update threshold, a location update is generated. AMM uses threshold bounds, called $upper$ and $lower\ threshold\ bounds$ to specify an appropriate update threshold value for each object. The objects which satisfy the condition of a LDCQ currently, or will satisfy the condition in the near future are assigned smaller update threshold values. Therefore, the locations of such objects are monitored more closely to get precise answers to the related LDCQs. More specifically, the update threshold of a moving object S is determined based on the $begin\ time$ of the object, by using the following formula:

$$H - (H - L) \times e^{-\delta t} \tag{1}$$

where

H : Upper Update Threshold bound,

L : Lower Update Threshold bound, and

δt : $begin\ time$ of object S - $current\ time$.

A moving object might be in the answer set of more than one LDCQ. In that case the update threshold of the object is set to be the minimum of all the thresholds computed from all the related LDCQs.

As an example, suppose that a mobile client has submitted a LDCQ, and in the first evaluation of the query at current time 20, the following answer set is generated:

$\{< S_1, 30, 40 >, < S_2, 25, 35 >, < S_3, 50, 60 >, < S_4, 20, 30 >\}$

In Figure 1, the update threshold values of mobile objects S_1, S_2, S_3 and S_4 determined by using Equation (1) are shown.

In the update threshold formula (1), an exponential distribution is assumed for illustration. It is possible to adapt different functions for different systems with the consideration of the location update cost and the cost of missing the information to the clients.

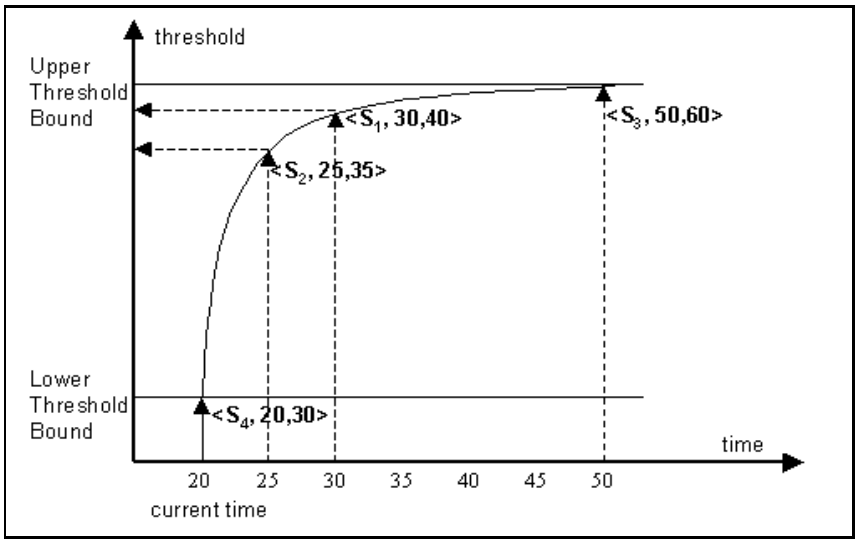

Figure 1: Generation of update thresholds in AMM.

3 A Criticality-Based Method for Generating Location Updates

If a query result provides false information, either in $begin/end\ times$ of objects, or in value, the query result is considered to be $incorrect$. In [La01], the correctness of a result is defined based on the $begin\ time$ of the result. The $actual\ begin\ time$ of an object is defined as the time when the object starts to satisfy the conditions of a query. The actual begin time of an object may be different from the $begin\ time$ of the corresponding tuple for the same query since the database may contain outdated information about the current location of the object. A mobile client may observe incorrect result if:

(1) the actual begin time of an object for a query is earlier than the $begin\ time$ of the result tuple produced for that object (this problem is called $missed\ information$); or

(2) the actual begin time of an object for a query is later than the $begin\ time$ of the result tuple produced for that object (this problem is called $false\ information$).

In the first case, although the moving object satisfies the query, the requesting client is not aware of that situation (Figure 2). In the second case, the requesting mobile client is informed that an object meets the condition of its query but actually it does not. Message transmission delays and outdated database state are the major causes for those incorrect results.

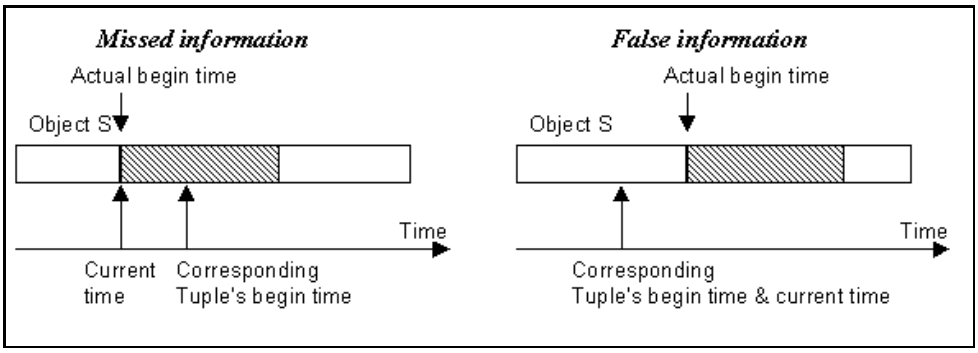

Figure 2: Missed and false information.

3.1 Motivation

Some queries in mobile computing systems can be characterized to be more critical than the others, due to the lower levels of tolerance to the inaccurate query results to mobile clients. In order to cope with the criticality of the underlying application (e.g., ambulatory service of a hospital), timely and accurate results should be provided to such queries.

We propose a location update generation method, called $Categorized\ Adaptive\ Monitoring\ Method$ (CAMM), for the moving objects with the aim of maintaining high levels of accuracy for the results of LDCQs that are characterized by high levels of criticality. In this method, n criticality levels are defined for the clients generating LDCQs. A client may generate LDCQs with one of the criticality categories of its own criticality level. For example, ambulatory service of a hospital may operate in a mobile environment with criticality level n=4 and may generate a LDCQ with a criticality category of either 1, 2, 3 or 4 in that level. Update threshold values of moving objects in the answer set of a query are determined by the criticality category of the query. Our emphasis in CAMM is providing relatively timely and correct results to LDCQs considering their criticality categories.

3.2 Determination of Location Update Thresholds

In implementing our method CAMM, we assume that moving objects generate updates to report their current locations to the database server which maintains a database containing the data items issued with the moving objects (including their location information). Each mobile client may generate LDCQs with different criticality categories. We have defined n=4 levels of criticality for the mobile clients generating LDCQs. The result set produced for the LDCQ of a client in criticality level 1 is exactly same as that produced by AMM. A client with criticality level 2 may generate a LDCQ either with a criticality category 1 or criticality category 2. CAMM can generate more accurate and timely results to a LDCQ with criticality category 2 than that to a LDCQ with criticality category 1.

Similar to AMM, CAMM also uses location update threshold bounds for determining update thresholds of moving objects. If the deviation between the actual and computed values of the location of an object is greater than the update threshold of the object, then a location update is generated for that object. Location update threshold of a moving object S is determined by the following formula:

$$C_{up} - (C_{up} - C_{lw}) \times e^{-\delta t} \tag{2}$$

where,

C_{up}: Criticality Category Upper bound

C_{lw}: Criticality Category Lower bound

δt : $begin\ time$ of object S - $current\ time$

$$C_{up} = H - (C_c - 1)\left(\frac{H - L}{C_l}\right) \tag{3}$$

$$C_{lw} = L \tag{4}$$

where,

H : Upper Update Threshold bound

L : Lower Update Threshold bound

C_c : Criticality Category of the query generated

C_l : Criticality Level of the client generating the query

The idea behind this formulation is to map the threshold bounds interval of AMM to n equal subintervals for n different criticality categories assigned to the queries submitted by a mobile client with criticality level n. As the criticality category increases, the level of the subinterval bounding the update thresholds becomes lower.

As an example, suppose that a client with criticality level C_l=4 has submitted a LDCQ with criticality category C_c=2. In the first evaluation of the query at time t=10, the tuples and update thresholds generated for moving objects S_1, S_2 and S_3 are shown in Figure 3.

4 Performance Experiments

We have designed and implemented a detailed simulation model to study the performance of the proposed method, CAMM, as compared to AMM which was already proven to perform better than the well-known approaches for location update generation in mobile systems [La01]. The simulation program was written in CSIM18 [Sc86]. The simulation model and the performance results are presented in the following subsections.

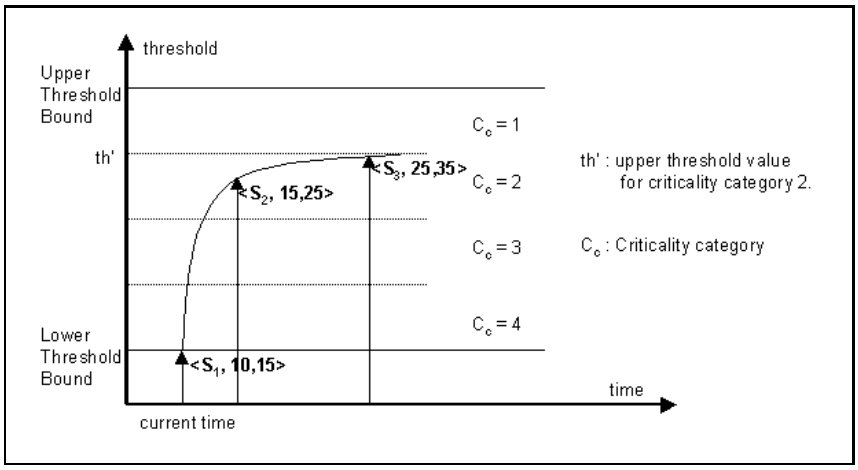

Figure 3: An example for the generation of location update thresholds in CAMM.

4.1 Simulation Model

We have extended the performance model that we used in our previous work [GU00, La01] to support modeling of processing LDCQs with different criticality categories. Our simulation model consists of three basic components:

- Server Model
- Wireless Communication Network Manager
- Mobile Client Model

The server model has two modules corresponding to a query processor and a resource manager. The query processor processes queries received from clients through the wireless communication network. It also updates the answer set of LDCQs whenever a change occurs in location of related objects. The resource manager is responsible for scheduling CPU and database accesses. The wireless communication network manager module carries all the traffic between the server and the client.

The mobile client model consists of three components: a resource manager, a query generator, and an update generator. The resource manager models the CPU scheduling at the client machine for processing queries and transmitting them to the server. The query generator generates queries, while the update generator generates location updates of moving objects using the update threshold values determined by CAMM.

Key parameters of our simulation model are listed in Table 1 together with their default values used in experiments. The values of the simulation parameters were chosen so as to be comparable to the previous related simulation studies such as [GU00, La01]. It was not intended to simulate a specific application; instead the parameter values were chosen to yield a communication and query processing load high enough to observe the differences

PARAMETER	VALUE
Number of moving objects	200
Number of objects which may generate LDCQs	50
Maximum number of objects that can satisfy a query	20
Maximum number of criticality levels	4
Lifetime of a LDCQ	240 - 360 sec
Mean think time between queries	1000 sec
Message transmission time	0.1 - 0.2 sec
Time for processing a tuple	0.05 - 0.1 sec
Lower update threshold limit	5 sec
Upper update threshold limit	40 sec

Table 1: Default parameter settings

between the performances of location update methods in providing correct information to LDCQs. In our experiments, we are concerned with performance trends rather than exact performance predictions.

Criticality level of a client is chosen randomly from the set of all possible criticality levels. Assignment of criticality category to a query submitted by a client with criticality level i again follows a uniform distribution; i.e., the criticality of the query is chosen randomly from the set $\{1, ..., i\}$.

A number of simulation experiments have been conducted to study the performance of CAMM as a function of $incorrect\ information\ rate$ (IIR) which is the number of occurrences of false and missed information over the total number of tuples generated. As our main performance measure, IIR indicates the capability of the system in providing correct information to LDCQs generated by mobile clients.

4.2 Evaluation of the Impact of Location Update Threshold

We first investigate the performance of our location update method CAMM under varying values of the lower limit of location update threshold by changing the corresponding parameter value from 5 seconds to 40 seconds. Figure 4 presents the incorrect information rate (IIR) results obtained for both AMM and CAMM. It can be observed from the figure that the performance of CAMM is better, in general, than that of AMM, for various criticality levels tried with CAMM. All of the curves displayed are V-shaped, and this situation can be explained as follows. The poor performance obtained with very small threshold values is due to the heavy update workload. Obviously, under such high levels of update workload, most of the system resources are devoted to process the updates from moving objects due to small threshold values and the system is not able to generate timely response to the continuous queries from mobile clients. Consistent with our expectation, this situation causes an increase in incorrect (i.e., missed and false) information rate. When

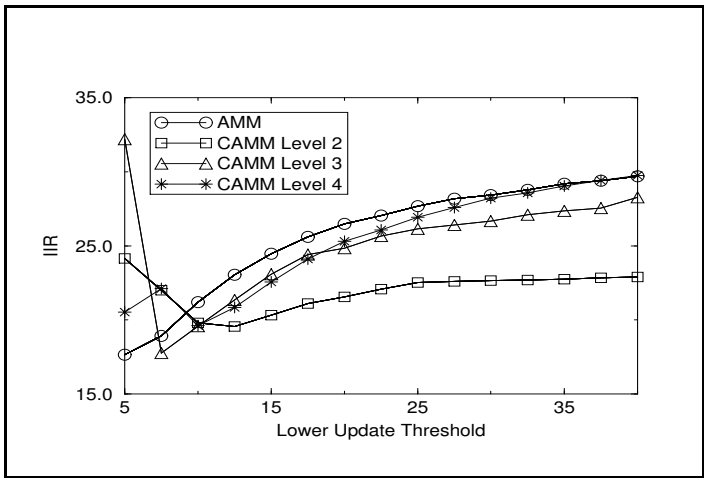

Figure 4: Incorrect Information Rate vs lower limit of location update threshold

a considerably large threshold value is used, on the other hand, moving objects update their location with larger time intervals and this also causes the probability of providing incorrect information to mobile clients to increase. In general, the best performance in terms of IIR is achieved when CAMM method with criticality level 2 is employed.

In Figure 5, IIR evaluation of criticality level 2 is provided to show the difference between the performance of category 1 queries and category 2 queries (which are more critical). Obviously, as expected, criticality category 1, which has higher upper threshold limit, produces higher IIR values than those of criticality category 2. So, more accurate results are provided to more critical category 2 queries.

4.3 Evaluation of the Impact of Continuous Query Lifetime

In the experiments of this section, we investigate the performance impact of the lifetime of LDCQs as the lower limit of update threshold is varied. We conducted the first experiment by reducing the lower and upper bounds of query lifetime to 60 and 240 seconds, respectively. In the second experiment, these bounds were increased to 240 and 600 seconds, respectively. The results obtained for CAMM with criticality level 2 are displayed in Figure 6. The results for short and long duration queries are presented together with those obtained with default settings of the lower and upper bounds of query lifetime; i.e., 240 and 360 seconds, respectively. As expected, shorter query lifetime leads to lower IIR results due to the reduced system workload. As the workload decreases, the system resources are devoted to generate more timely response to the continuous queries from mobile clients.

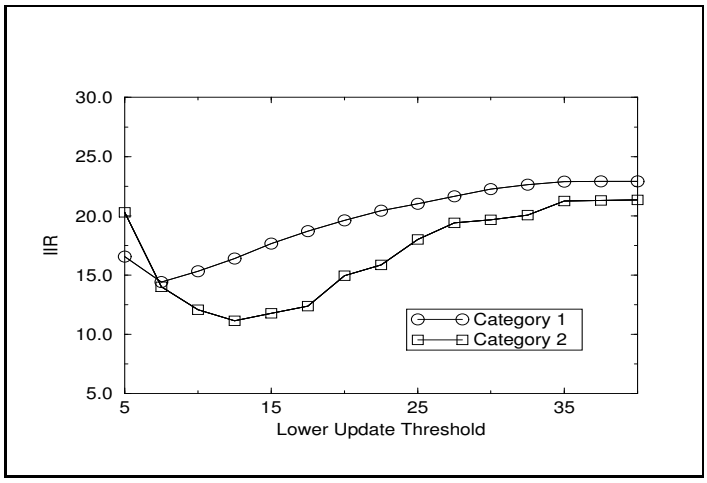

Figure 5: Incorrect Information Rate values for criticality level 2 queries in two different categories

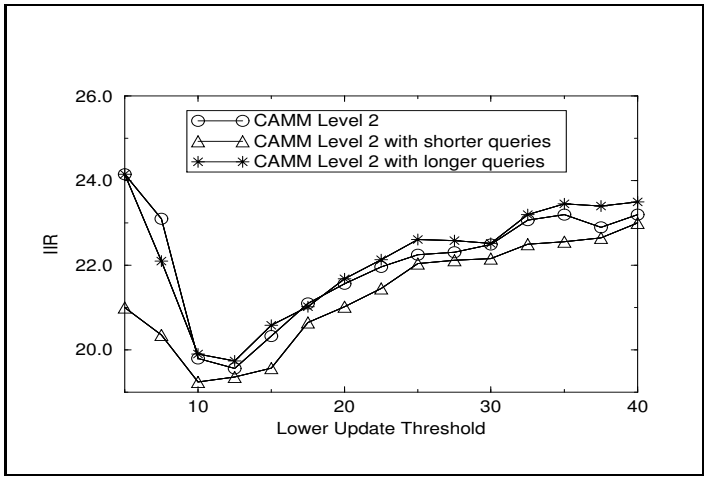

Figure 6: Incorrect Information Rate results obtained with different settings of continuous query lifetime

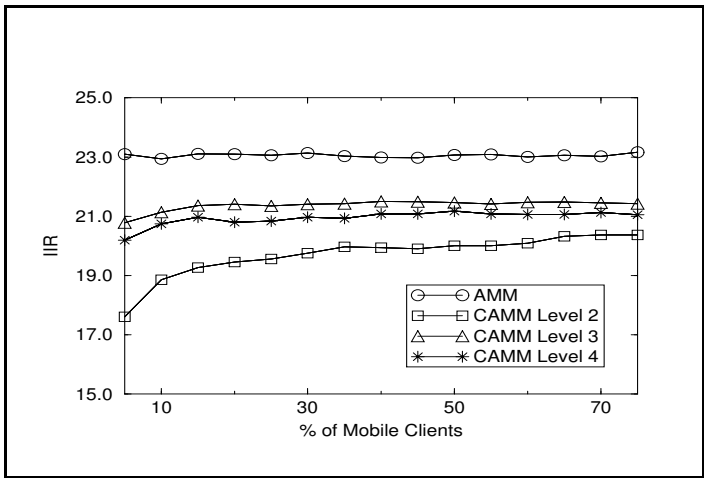

Figure 7: Incorrect information rate vs the percentage of mobile clients over the total number of moving objects

4.4 Evaluation of the Impact of Number of Mobile Clients

We also investigated the performance impact of the number of mobile clients which may generate LDCQs. Figure 7 presents the IIR results obtained when the minimum location update threshold is fixed at 12.5 seconds and the percentage of the number of mobile clients within all mobile objects is varied from 5% to 75%.

Our observations with the previous experiments are also valid in this experiment. The best results in terms of IIR are obtained with the criticality level 2 of CAMM. Although CAMM with any criticality level produces a bit higher IIR with larger number of mobile clients, it still beats AMM even with the largest ratios of mobile clients over the total number of moving objects. The relative performance of AMM and different criticality levels of CAMM is not affected by the number of mobile clients.

5 Conclusions

In this paper, we have presented a new method, called Categorized Adaptive Monitoring Method (CAMM), for the generation of location updates in mobile computing systems. The method aims to increase accuracy and timeliness of the results of location dependent continuous queries, as the criticality of the queries increases. It enables the system to have criticality levels for both the clients which can submit queries, and the queries submitted, based on the requirement of the level of accuracy for query results. It was shown through performance experiments that a considerable improvement in the performance in terms of incorrect information rate is possible with CAMM, especially when only two levels of criticality are considered.

References

[AFZ96] Acharya, S.; Franklin, M.; Zdonik, S.: Disseminating Updates on Broadcast Disks. Proc. International Conference on Very Large Databases, 1996, pp.354-365.

[BJ96] Bukhres, O.; Jing, J.: Analysis of Adaptive Caching Algorithms in Mobile Environments. Information Sciences, vol.95, no.1, 1996, pp.1-27.

[Da97] Datta, A. et. al.: Adaptive Broadcast Protocol to Support Power Conservant Retrieval by Mobile Users. Proc. International Conference on Data Engineering, 1997, pp.124-133.

[DK98] Dunham, M.H.; Kumar, V.: Location Dependent Data and its Management in Mobile Databases. Proc. International Workshop of Database and Expert Systems Applications, 1998, pp.414-419.

[FR98] Fernandez, J.; Ramamritham, K.: Adaptive Dissemination of Data in Real-Time Asymmetric Communication Environments. Proc. Euromicro Conference on Real-Time Systems, 1998, pp.195-203.

[GU00] Gök, H. G.; Ulusoy, Ö.: Transmission of Continuous Query Results in Mobile Computing Systems. Information Sciences, vol.125, no.1-4, 2000, pp.37-63.

[IVB97] Imielinski, T.; Viswanathan, S.; Badrinath, B.R.: Data on Air: Organization and Access. IEEE Transactions on Knowledge and Data Engineering, vol.9, no.3, 1997, pp.353-372.

[KGT99] Kollios, G.; Gunopulos, D.; Tsotras, V.J.: On Indexing Mobile Objects. Proc. Principles of Database Systems, 1999, pp.261-272.

[KZ98] Kottkamp, H.; Zukunft, O.: Location-Aware Query Processing in Mobile Database Systems. Proc. ACM Symposium on Applied Computing, 1998, pp.416-423.

[La01] Lam, K. Y. et. al.: An Efficient Method for Generating Location Updates for Processing of Location-Dependent Continuous Queries. Proc. International Conference on Database Systems for Advanced Applications, 2001, pp.218-225.

[RD00] Ren, Q.; Dunham, M.H.: Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. Proc. International Conference on Mobile Computing and Networking, 2000, pp.210-221.

[Sc86] Schwetman, H. D.: CSIM: A C-Based, Process-Oriented Simulation Language. Proc. Winter Simulation Conference, 1996, pp.387-396.

[Si97] Sistla, A. P. et. al.: Modeling and Querying Moving Objects. Proc. International Conference on Data Engineering, 1997, pp.422-432.

[Si98] Sistla, A. P. et. al.: Querying the Uncertain Position of Moving Objects. Temporal Databases: Research and Practice. Lecture Notes in Computer Science (Springer Verlag), vol.1399, 1998, pp.310-337.

[SDK01] Seydim, A.Y.; Dunham, M.H.; Kumar, V.: Location Dependent Query Processing. Proc. ACM International Workshop on Data Engineering for Wireless and Mobile Access, 2001, pp.47-53.

[TUW98] Tayeb, J.; Ulusoy, Ö., Wolfson, O.: A Quadtree Based Dynamic Attribute Indexing Method. The Computer Journal, vol.41, no.3, 1998, pp.185-200.

[Wo97] Wolfson, O. et. al.: Location Management in Moving Objects Databases. Proc. International Workshop on Satellite-Based Information Systems, 1997, pp.7-13.

[Wo99] Wolfson, O. et. al.: Updating and Querying Databases that Track Mobile Units. Distributed and Parallel Databases, vol.7, no.3, 1999, pp.257-287.

[WL01] Wong, V. W. S.; Leung, V. C. M.: An Adaptive Distance-Based Location Update Algorithm for Next Generation PCS Networks. IEEE Journal on Selected Areas in Communications, vol.19, no.10, 2001, pp.1942-1952.