PROCESSING OF CONTINUOUS QUERIES FROM MOVING OBJECTS IN MOBILE COMPUTING SYSTEMS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATION SCIENCE
AND THE INSTUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By

Hüseyin Gökmen Gök

January, 1999

PROCESSING OF CONTINUOUS QUERIES FROM MOVING OBJECTS IN MOBILE COMPUTING SYSTEMS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hüseyin Gökmen Gök

January, 1999

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Özgür Ulusoy(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Tuğrul Dayar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Uğur Güdükbay

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

PROCESSING OF CONTINUOUS QUERIES FROM MOVING OBJECTS IN MOBILE COMPUTING SYSTEMS

Hüseyin Gökmen Gök
M.S. in Computer Engineering and Information Science
Supervisor: Assoc. Prof. Dr. Özgür Ulusoy
January, 1999

Recent advances in computer hardware technology and wireless communication networks have led to the emergence of mobile computing systems. In a mobile computing environment, a user with a wireless connection to the information network can access data via submitting queries to the data server. Since the mobility is the most distinguishing feature of the mobile computing paradigm, location becomes an important piece of information for the so called *location-dependent queries* where the answer to a query depends on the current location of the user who issued the query. A location-dependent query submitted by a mobile user can become more difficult to process when it is submitted as a *continuous query* for which the answer changes as the user moves. The answer to a location-dependent continuous query is a set that consists of tuples $< S, begin, end >$ indicating that object S is the answer of the query from time *begin* to time *end*. Once the tuples in the answer set are determined, the next step is to determine when to send these tuples to the user. The transmission time of the tuples is critical in the sense that it can affect the communication overhead imposed on the wireless network and the availability of tuples in case of disconnections. In this thesis, we propose three tuple transmission approaches that determine the transmission time of a tuple in the answer set of a location-dependent continuous query. We also design and implement a simulation model to compare the performance of the proposed tuple transmission approaches under different settings of environmental parameters.

Key words: Mobile Computing, Mobile Database Systems, Location-Dependent Queries, Continuous Queries, Simulation.

ÖZET

MOBİL İLETİŞİM ORTAMLARINDA HAREKETLİ KULLANICILARDAN GÖNDERİLEN SÜREKLİ SORGULARIN İŞLENMESİ

Hüseyin Gökmen Gök
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Özgür Ulusoy
Ocak, 1999

Bilgisayar donanımı ve telsiz iletişim ağı teknolojilerindeki gelişmeler mobil iletişim ortamlarının gelişmesine yolaçtı. Mobil iletişim ortamlarında, bilgi ağına telsiz bağlantısı olan kullanıcılar, veri sunucusuna sorgular göndererek veriye ulaşırlar. Mobil iletişim ortamlarında kullanıcıların hareketli olması nedeniyle, kullanıcıların konumları *konuma-dayalı sorgular* açısından önemli bir bilgidir. Konuma-dayalı sorgular *sürekli sorgular* haline getirildiğinde daha da karmaşıklaşırlar, çünkü sorgunun cevabı mobil kullanıcının hareket etmesi nedeniyle sürekli değişir. Konuma-dayalı sürekli bir sorgunun cevap kümesi $< S$, *başlangıç, bitiş* $>$ gibi elemanlardan oluşur ve her bir eleman S nesnesinin *başlangıç* ve *bitiş* süreleri arasında sorgunun cevabı olduğu anlamına gelir. Cevap kümesindeki elemanların belirlenmesinden bir sonraki aşama bu elemanların mobil kullanıcıya ne zaman gönderileceğidir. Bu zamanlama telsiz ağ üzerindeki iletişim yükünü ve bağlantı kopukluğu durumunda sorgu cevabının ne kadarının kullanıcıya gönderilmiş olduğunu etkilemesi açısından kritiktir. Bu tezde konuma-dayalı sürekli sorguların cevap kümesindeki elemanların mobil kullanıcılara gönderiliş zamanını belirleyen üç değişik metot önerilmektedir. Bunun yanında, önerilen metotların değişik ortamlardaki performanslarını karşılaştırabilmek amacıyla bir simülasyon modeli tasarlanmış ve gerçekleştirilmiştir.

Anahtar kelimeler: Mobil İletişim, Mobil Veritabanı Sistemleri, Konuma Dayalı Sorgular, Sürekli Sorgular, Simülasyon.

To my family

ACKNOWLEDGMENTS

I am very grateful to my supervisor Assoc. Prof. Dr. Özgür Ulusoy for his invaluable guidance and motivating support during this study. His instruction will be the closest and most important reference in my future research.

I would also like to thank John Wu for the discussions about porting CSIM to Linux, Şirvan Yıldız who shared many good ideas with me, thereby contributing valuable suggestions, Yücel Saygın for the words of encouragement, Halime Sultan for the great motivation throughout the whole study, and my family for giving me the patient understanding and love without which this study could not have been completed.

Finally, I would like to thank my committee members Asst. Prof. Dr. Tuğrul Dayar and Asst. Prof. Dr. Uğur Güdükbay for their comments, and everybody who has in some way contributed to this study.

Contents

1 Introduction **1**

2 Related Work **7**

3 Background and Motivation **10**

4 Tuple Transmission Approaches **15**

4.1 Immediate Transmission (IT) Approach 15

4.2 Delayed Transmission (DT) Approach 16

4.3 Periodic Transmission (PT) Approach 16

4.4 Adaptive Periodic Transmission (APT) Approach 17

4.5 Mixed Transmission (MT) Approach 18

5 Simulation Model **21**

5.1 Mobile Client Model 22

5.2 Wireless Network Manager 25

5.3 Server Model 25

6 Experiments and Results **30**

6.1 System Performance Metrics 30

6.2 Parameter Settings 31

6.3 The Base Experiment 33

6.3.1 Evaluation of the Impact of Query Duration 36

6.3.2 Evaluation of the Impact of Disconnection Period 37

6.4 Evaluation of the Impact of Hotspots 38

6.4.1 Evaluation of the Impact of Query Duration 42

6.4.2 Evaluation of the Impact of Disconnection Period 42

6.4.3 Evaluation of the Impact of Query Hotspots 44

7 Conclusions and Future Work **47**

List of Figures

1.1 System Model of a Mobile Computing Environment. 3

3.1 Basic Communication Between an MH and an MSS. 11

3.2 Possible Effects of an Explicit Update. 12

5.1 The Simulation Model. 22

5.2 Mobile Client Model. 23

5.3 Server Model. 26

6.1 Average Number of Bits Transmitted vs Data Update Rate. 34

6.2 Average Number of Control Messages vs Data Update Rate. . . 34

6.3 Average Number of Retransmitted Tuples per CQ vs Data Update Rate. 35

6.4 Availability of Tuples vs Data Update Rate. 35

6.5 Average Number of Retransmitted Tuples vs Maximum Query Duration. 37

6.6 Average Number of Bits Transmitted vs Maximum Query Duration. 38

6.7 Availability of Tuples vs Disconnection Period. 39

6.8 Average Number of Bits Transmitted vs Data Update Rate. 40

6.9 Average Number of Retransmitted Tuples vs Data Update Rate. 40

6.10 Average Number of Control Messages vs Data Update Rate. . . 41

6.11 Availability of Tuples vs Data Update Rate. 41

6.12 Average Number of Bits Transmitted vs Maximum Query Duration. 43

6.13 Availability of Tuples vs Disconnection Period. 44

6.14 Average Number of Bits Transmitted vs Data Update Rate. 45

6.15 Average Number of Retransmitted Tuples per CQ vs Data Update Rate. 45

List of Tables

5.1 Mobile Client Model Parameters 24

5.2 Wireless Network Manager Parameters 25

5.3 Server Model Parameters 27

6.1 Parameter Settings for The Base Experiment 32

6.2 Parameter Settings 39

Chapter 1

Introduction

Recent advances in computer hardware technology and wireless communication networks have led to the emergence of mobile computing systems [PB93, FZ94]. In a mobile computing environment, a user with a wireless connection to the information network does not require to maintain a fixed position in the network [Chr93, WC95].

Mobility has opened up new areas of research in networking and distributed database management systems because traditional techniques developed for those systems have been based on the assumption that the location of the hosts and the connections among them do not change. In a mobile computing environment, users carrying portable computers wish to maintain transparent network access through wireless links while they move from one place to another. It is expected that in the near future, millions of mobile users will make use of integrated voice, data, and image applications [PB94]. Therefore, the existing hardware and software systems need to be improved based on the features and the requirements of this new computing environment.

The principal features of mobile computing are: *wireless communication*, *mobility* and *portability* [FZ94]. Wireless communication is much more difficult than wired communication because the surrounding environment interacts with the signal introducing noise and echoes [FZ94]. Some of the implications of using wireless communication are: susceptibility to disconnection, highly variable

network conditions, and low bandwidth availability. It seems that the wireless network bandwidth will remain a major limitation and performance bottleneck for mobile system design in the near future [PB93, FZ94, WC97, SW98].

A mobile computer can be disconnected from the network intentionally or due to failures. It can also be possible to predict the disconnections. For example, a weak radio link, or a partially depleted battery may warn of disconnection possibility. Once disconnected, a mobile computer can later reconnect to the network but in environments with frequent disconnections, it is essential for the mobile computer to be able to operate in stand-alone mode during the disconnection period.

The ability to change location while connected to the network increases the volatility of some information. Certain data considered static for stationary computing becomes dynamic for mobile computing. For example, although a stationary computer can be configured statically to prefer the nearest server, a mobile computer needs a mechanism to determine which server to use. Mobility makes the location of the user a fast-changing data. Hence, processing of user queries depending on the location of the mobile user is an important issue that needs to be handled.

Mobile (portable) computers are to be carried by users, so their design must not be liberal in their use of space and power. Portability places pressure on the design of the mobile system in terms of both hardware and software design due to the requirements for the consideration of low power consumption, risk of data loss, and small surface area available for the user interface. Therefore, portability entails limited resources available on board to handle the dynamic mobile computing environment. As a result of that, it might be required to operate a mobile computer in the *doze* mode for conserving energy. During this mode of operation, the clock speed is reduced and no user computations are performed. The mobile computer waits in the doze mode until it receives a message from the rest of the network. Upon receipt of any such message, the mobile computer resumes its normal mode of execution.

A widely accepted mobile system model [PB93, WC95, BMM96, TKN96, WC97, PS97], as shown in Figure 1.1, consists of two distinct sets of entities:

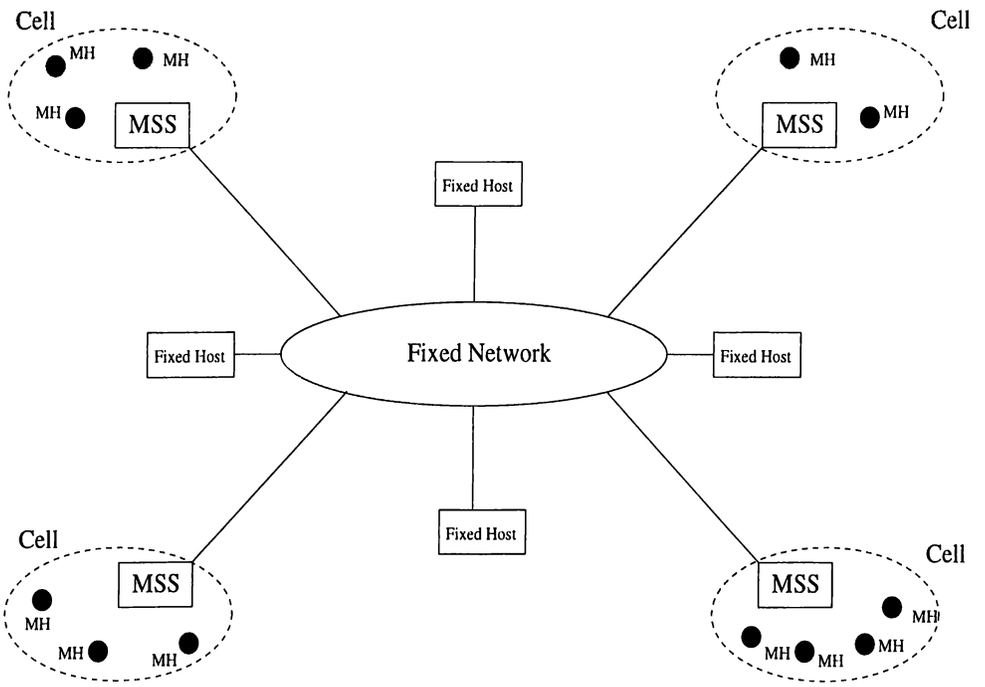

Figure 1.1: System Model of a Mobile Computing Environment.

mobile hosts and fixed hosts. A *mobile host* (MH) is able to move without losing its network connection. Some of the fixed hosts that are called *mobile support stations* (MSS) have the ability to communicate with mobile hosts via wireless network. A *cell* is a geographical coverage area under an MSS. Each MH is associated with an MSS (i.e., it belongs to the cell serviced by the MSS). An MH can directly communicate with an MSS if the MH is physically located within the cell serviced by the MSS. In order to communicate with an MH that is not in the same cell, the source MH contacts with its local MSS which forwards the message to the MSS of the target MH over the wired network. The receiving MSS then transmits the message over wireless network to the target MH.

When an MH is engaged in a data transfer, it is possible that it can move out of the coverage area of the local MSS. Unless the data transfer is passed on to the current cell of the MH, it will be lost. Therefore, the task of forwarding data between the static network and the MH must be transferred to the new cell's MSS. This process, called *hand-off*, is transparent to the user [PB93].

Since the mobility is the most distinguishing feature of the mobile computing

paradigm, location becomes an important piece of information for the so called *location-dependent queries* [SWCD97, SWCD98, TUW98, WXCJ98]. Consider a database representing information about moving objects and their position in addition to information about stationary objects. A typical query submitted to a hotel management system might be: "display motels (with room price and availability) that are within 5 miles of my position"; or in a battlefield a typical query submitted might be: "display the friendly tanks within 10 miles of my position". Such queries may be issued from a moving object (e.g., car of a mobile user) or from a stationary user. Consequently, the answer to a location-dependent query may depend on the location of the MH which issued the query and/or the locations of the objects represented in the database.

A location-dependent query can become more difficult to process when it is submitted as a *continuous query* (CQ) [SWCD97, SWCD98]. The driver querying the motels in the above example may request the answer to the query to be continuously updated so that he/she can find a motel with a reasonable price. It is clear that the answer to such a query changes with the car movement and continuously updating driver's location would impose a serious performance and wireless bandwidth overhead. Existing database management systems (DBMSs) are not well equipped to handle continuously changing data such as the position of moving objects, since the data is assumed to be constant unless it is explicitly modified. The position of a moving object changes continuously as a function of time. Hence, the answer to a CQ depends not only on the database contents but also on the time at which the query is issued.

In [SWCD97, SWCD98], a new data model called *Moving Objects Spatio-Temporal* (MOST) is proposed 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 O" in the MOST data model is different for time points t_1 and t_2 even if the value of the attribute specifying O's position has not been explicitly updated.

Consider again the CQ: "display motels within 5 miles of my position" issued by a person driving a car. When such a CQ 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 CQ 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 transmit the tuples in the answer set of the CQ to the MH. There are two basic approaches introduced in [SWCD97] to transmit the tuples to the MH: *Immediate Transmission* (IT) and *Delayed Transmission* (DT). In the IT approach, the whole answer set is transmitted immediately after being computed. In the DT approach, each tuple $< S, begin, end >$ is transmitted to the mobile host at time *begin.*

In this study, we present three new approaches for the transmission of the tuples in the answer set of a location-dependent CQ. The first approach called *Periodic Transmission* (PT) transmits the tuples in the answer set periodically. At each w time units, this method transmits all the tuples $< S, begin, end >$ satisfying the condition $t \leq begin < t + w$ where t is the current time and w is the size of the time window. In the second approach which we call *Adaptive Periodic Transmission* (APT), as an extension to the first approach, w is dynamically adjusted according to the communication overhead changing due to environmental parameters such as data update rate, disconnection frequency, and disconnection period. The final approach, called *Mixed Transmission* (MT), differs from the first two approaches in that data objects are partitioned into two groups: one consisting of "hot" objects of updates and the other of "cold" objects of updates. This approach transmits the "hot" tuples as in APT and "cold" tuples as in IT.

We have implemented a simulation model of a mobile client-server system that supports processing of CQs issued by MHs over the database of moving objects. The simulation model is used to study the performance of the proposed approaches in terms of the communication overhead from the server to the MH and also to investigate performance enhancements of these approaches over the basic schemes provided in [SWCD97, SWCD98].

The remainder of this thesis is organized as follows. Chapter 2 discusses the related work. Chapter 3 presents the background and the motivation for our work. Chapter 4 describes the approaches provided to determine the transmission time of the tuples in the answer set of location-dependent CQs. Chapter 5

presents the simulation model used to evaluate the performance of the proposed approaches. Chapter 6 describes the experiments conducted and discusses the results obtained. Concluding remarks and the future work are presented in Chapter 7.

Chapter 2

Related Work

The field of mobile database systems has been a hot research topic during the last couple of years. A mobile computing environment can be characterized by frequent disconnections of MHs, significant limitations of bandwidth and power, resource restrictions, and fast changing locations. All such characteristics associated with mobile systems make traditional techniques used in distributed computing systems inadequate and raise new challenging research problems.

There exist a considerable a number of papers discussing general issues and research challenges related to mobility. The new challenges in mobile data management are identified and their technical significance is investigated in [IB93, IB94]. [DHB97] focuses on the differences between data management solutions in a mobile computing environment and those in a distributed database environment. The impact of mobility on current software systems is discussed in [PB93]. Fundamental software design problems particular to the mobile computing environment are addressed in [FZ94]. A general architecture for a mobile information system is described in [PB94].

There has recently been much research concerning transaction processing strategies for the mobile computing environment. Distributed transaction processing issues are reexamined to account for the requirements of the mobile

environment and an algorithm is proposed in [EJB95] to coordinate the execution of the operations of a transaction running at different servers. That paper also provides a comparison between the proposed algorithm and existing solutions that use the two-phase-commit protocol. [WC97] proposes a transaction processing system that supports disconnections. Movement behavior of the MHs is captured in a transaction model presented in [DHB97]. [Chr93] proposes an open-nested transaction model for the mobile computing environment. Employing semantic knowledge to achieve a high degree of concurrency and to simplify recovery in the presence of failures are discussed in [WC95].

Location management of MHs has also been studied intensively. Distributed location management schemes are provided in [AP95, RB95] to keep track of the location of an MH. Another distributed location management strategy with fast location update and query, and load balancing among location servers is proposed in [PS97]. [TKN96] combines the problem of location management and query processing. It discusses several strategies for efficient processing of queries to obtain the location of an MH, queries to determine whether an MH is currently active, and queries to obtain information from an MH. Query optimization considering both resource utilization and power consumption at MHs is discussed in [AG93, GA93].

The problems associated with the indexing of the dynamic attributes (such as location) in a mobile database system are addressed in [TUW98]. 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. Indexing the position of moving objects as a dynamic attribute for location-dependent queries is exclusively discussed in [TUW]. A solution with a simple algorithm evolving the index through time with optimal overhead is proposed.

Development of caching strategies to reduce the communication cost has attracted the database community since communication in a mobile computing environment is expensive. Some caching strategies are introduced in [BI94]. The performance of these algorithms and the impact of MH's disconnection times on these strategies are evaluated. [WL95] proposes a caching strategy to maintain cache consistency so that locks are not required for read-only

transactions. The concept of "air-storage" by treating the wireless media as a layer of cache storage is considered in [LS97]. Another study [BI93] broadcasts the timestamps of the latest changes in items as an invalidation mechanism.

The problem of cache invalidation in mobile environments is addressed in detail in [BJ96]. The basic idea behind the APS approach presented in our thesis was inspired from the adaptive caching algorithm introduced in that paper. However, our context of adaptiveness is completely different. The problem we address is the determination of transmission times of the tuples in the answer set of a location dependent CQ, rather than the problem of cache invalidation. In order to adapt to the environmental parameters, the APS approach focuses on the overhead caused by the control messages and the retransmissions whereas the adaptive caching algorithm in [BJ96] deals with the overhead of the false cache invalidation.

The most relevant work to ours is the one presented in a series of papers [SWCD97, SWCD98, WXCJ98, WSCY]. Issues related to moving objects databases such as indexing, location updates of moving objects, modeling, and querying moving objects are exclusively addressed in these papers. A new data model (MOST) is proposed to model moving objects. Future Temporal Logic (FTL) is proposed as the query language for the MOST data model. An algorithm for processing FTL queries in the MOST data model is also provided. Two basic approaches are provided for the problem of when to transmit the tuples in the answer set of a CQ.

Chapter 3

Background and Motivation

According to the *Moving Objects Spatio-Temporal* (MOST) data model proposed in [SWCD97, SWCD98], a *static* attribute of a database object is an attribute that changes only when an explicit update is applied on it; in contrast a *dynamic* attribute of a database object changes over time according to a certain function even it is not explicitly updated. For example, each of the x, y coordinates of a moving object that specify the position of the object in two dimensional space, is a dynamic attribute. In the MOST data model, a dynamic attribute A is represented by 3 subattributes:

1. $A.value$
2. $A.updatetime$
3. $A.function$

$A.function$ is a function of time (t) which has value 0 at $t = 0$. At time $A.updatetime$ the value of A is $A.value$. Thus, until the next update time, the value of A at time $A.time + t_0$ is given by $A.value + A.function(t_0)$. Unlike the traditional database systems where the same value for the attribute is returned unless the attribute has not been explicitly modified, in the MOST data model the value of a dynamic attribute depends on the time at which it is queried.

An explicit update of a dynamic attribute changes the value of the above 3 subattributes that represent the position of a moving object. Therefore, the

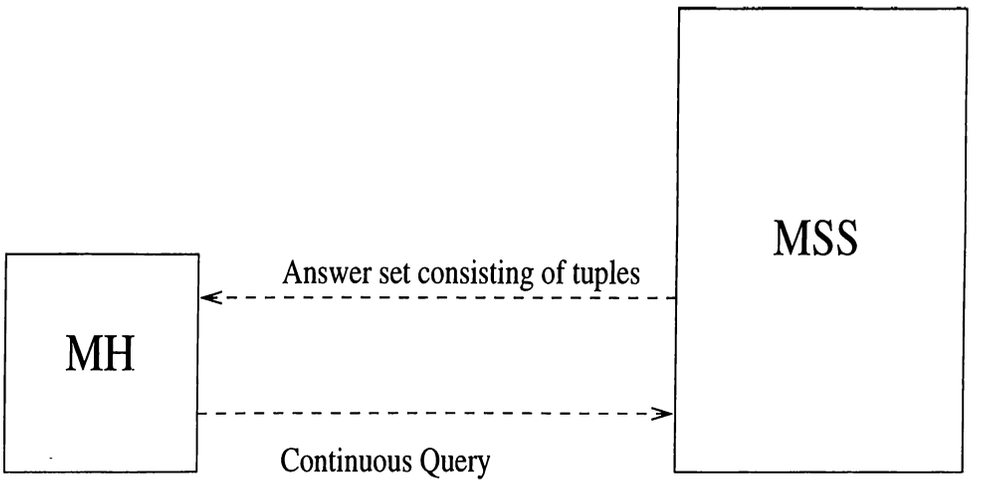

Figure 3.1: Basic Communication Between an MH and an MSS.

attributes representing the position of a moving object can remain unchanged, while the position of the moving object changes. In the MOST data model, the database implicitly represents future states such as the future positions of moving objects, therefore queries referring to the future rather than the current state of the system can be answered.

Consider again the query: "display motels within 5 miles of my position" issued by a moving object. Recall that the answer to this query has to be continuously updated (at least until a motel with a reasonable price is found). Continuously evaluating such a query would be very inefficient. The query processing algorithm proposed in [SWCD97, SWCD98] evaluates the query once and returns a set of tuples. Figure 3.1 illustrates the basic communication between an MH and an MSS. For an issued CQ, the answer set consists of tuples $< S, begin, end >$ which means that object S satisfies the query between the times *begin* and *end*. In other words the MH will display object S on its screen between the times *begin* and *end*.

The work of [SWCD97, SWCD98] considers a centralized DBMS equipped with the MOST capability. Once the tuples to be transmitted to the MH are determined, the next step is to determine when to transmit all these tuples. In this study, the problem we attack is determination of the time to transmit the tuples in the answer set of an issued location-dependent CQ. The selection among the choices of transmitting all the tuples together at the time they are

I.
$Ans(CQ_1) = \{< S, 3, 10 >\}$
$Ans(CQ_2) = \{< O, 7, 9 >\}$
(Initial answer sets)

II.
$Ans(CQ_1) = \{< S, 5, 13 >\}$
$Ans(CQ_2) = \{< O, 7, 9 >\}$
(Answer sets after an update on object S)

III.
$Ans(CQ_1) = \{\}$
$Ans(CQ_2) = \{< O, 7, 9 >, < S, 4, 7 >\}$
(Answer sets after another update on object S)

Figure 3.2: Possible Effects of an Explicit Update.

determined, or delaying the transmission of a tuple until its *begin* time, or transmitting the tuples periodically can affect the MH which issued the CQ in terms of both communication cost and power consumption.

There are two basic dimensions of the communication overhead regarding the transmission of the tuples in the answer set of a CQ:

1. Control Message Overhead: According to the point to point communication paradigm [SWD+96], a message to be transmitted is appended to a fixed size control message.

2. Tuple Retransmission Overhead: An explicit update to an object in the database may change the tuples referring to the updated object as shown in Figure 3.2. The same object may satisfy the query but *begin* and/or *end* attribute of the tuple may change (Figure 3.2, I and II). It is also possible that a tuple referring to the updated object may no longer satisfy the query (Figure 3.2, II and III), and/or a new object may satisfy one or more of the active queries that it did not satisfy previously (Figure 3.2, II and III).

Suppose that the subattributes representing the position of a moving object S are explicitly updated at time t_1 and the tuple $< S, begin, end >$ referring to S is updated accordingly (i.e., the tuple still satisfies the corresponding query). As far as the *begin* time of the tuple is concerned, there are two possible cases:

Case 1. $t_1 \leq begin$

Case 2. $t_1 > begin$

In the first case, a retransmission of the tuple to the corresponding MH is necessary only if the tuple was previously transmitted to the MH. In the second case, a retransmission is mandatory because the tuple must have been transmitted to the MH by the time *begin*.

We want to make it clear that various tuple transmission approaches may handle Case 1 differently because it is possible to transmit a tuple at anytime $t \leq begin$. In contrast, retransmission at the time of update cannot be avoided with any approach in Case 2. Therefore, from now on we limit the scope of the retransmissions to exclude the ones that are due to an explicit update at $t_1 > begin$.

In order to minimize the control message overhead, all tuples to be transmitted to the MH should be gathered and form a single message. This means that all tuples in the answer set are transmitted at anytime before the *begin* time of the tuple with the earliest *begin*. On the other hand, such a strategy increases the probability that the tuple will be retransmitted to the MH in case of an explicit update. In order to minimize the probability of retransmission of a tuple in case of an explicit update, the tuple should be transmitted by its *begin* time. However, in the worst case such a strategy will lead to a situation where each tuple is appended to a control message. It is clear that the efforts for reducing the control message overhead increases the retransmission overhead and vice versa.

Given the same set of tuples as the answer to a CQ, different tuple transmission strategies will lead to different number of control messages and retransmissions. This means that different amount of communication overhead is involved with each strategy. Therefore, the tuple transmission time is critical

especially for the applications where message transmission service is charged a fixed amount of money per byte basis. For example, RAM Mobile Data Corporation charges a minimum of 4 cents per message, with the exact cost depending on the size of the message [WSCY]. Given a set of tuples as the answer to a CQ, different tuple transmission approaches produce bills with different amounts.

Underlying tuple transmission approach also affects the duration the MH operates in doze (energy saving) mode. CQs are processed entirely by the server. That is why, the number of transmissions and the total time the MH spends listening to the communication channel must be minimized in order to minimize the energy spent by the MH. Energy preservation is critical because MHs have limited battery capacity, two or three hours under normal use, which is expected to increase only 20% over the next 10 years [PB94, IB94]

Given the same set of tuples as the answer to a CQ, various tuple transmission strategies may differ in the ability to support the stand-alone working capability of an MH in case of disconnection. That is, when an MH is disconnected after receiving a number of tuples that are in the answer set of an issued CQ, it can continue displaying the received tuples during the disconnection period in the stand-alone mode (although the updates cannot be transmitted to it). The performance of tuple transmission approaches in terms of supporting the above ability may also be critical in some applications (e.g., in a battlefield).

Chapter 4

Tuple Transmission Approaches

In this chapter, we present the approaches which determine the transmission time of tuples in the answer set of a CQ issued by an MH. We also discuss the benefits and drawbacks of the approaches in terms of control message overhead, tuple retransmission overhead, and the handling of disconnection behavior.

4.1 Immediate Transmission (IT) Approach

According to the IT approach presented in [SWCD97, SWCD98], all the tuples that belong to the answer set of a CQ issued by an MH are transmitted at once at the time the query processing is finished. Upon receiving the answer set, the MH displays them on the screen accordingly. This approach has the following characteristics:

1. It minimizes the control message overhead. All tuples are gathered in a single message which also means a single control message.
2. When a tuple is changed due to an explicit update of an object after the query is processed, it has to be retransmitted.
3. In case the MH disconnects after sometime it has received the answer set of its query, it has the whole answer set.

4.2 Delayed Transmission (DT) Approach

According to the DT approach proposed in [SWCD97, SWCD98], a tuple $< S, begin, end >$ is transmitted to the MH at time *begin*. Upon receiving a tuple, the MH immediately displays it on the screen. This approach has the following characteristics:

1. It maximizes the control message overhead. Each tuple is appended to a control message and then transmitted.
2. The probability that a tuple has to be retransmitted in case of an explicit update to a database object, is minimized.
3. In case the MH disconnects after sometime it has started to receive the tuples in the answer of its CQ, it has the partial answer set.

4.3 Periodic Transmission (PT) Approach

PT is an intermediate approach lying between IT and DT. According to this approach, at each w time units, all the tuples $< S, begin, end >$ satisfying the condition $t \leq begin < t + w$ where t is the current time, are transmitted to the MH. We call w the window size which specifies the time interval containing the *begin* time of the tuples to be transmitted. This approach has the following characteristics:

1. The control message overhead is less than that of the DT approach but greater than that of the IT approach.
2. The probability that a tuple has to be retransmitted in case of an explicit update to a database object is less than it is in the IT approach but greater than it is in the DT approach.
3. In case the MH disconnects after sometime it has started to receive the tuples in the answer of its query, it has the partial answer set.

4.4 Adaptive Periodic Transmission (APT) Approach

The PT approach maintains a constant window size (w) for determining the tuple transmission times. The value of w affects both the control message overhead and the retransmission overhead. Large values of w reduces the control message overhead while increasing the retransmission overhead. Likewise, small values of w reduces the retransmission overhead while increasing the control message overhead.

Data update rate and the resulting overhead due to the retransmission of the updated tuples may vary during the execution of a mobile system. It might be appropriate to have a large w value in order to reduce the control message overhead when updates to the database objects are rare. Similarly, it might be appropriate to have a small w value in order to reduce the retransmission overhead when the updates are frequent. Taking into account the above facts, the APT approach adjusts w by evaluating the information about the relative overheads due to control messages and retransmissions. The period of adjustment of w is called the *evaluation period* of the window size.

The control message overhead is specified by the number of control message bits transmitted with the original tuples (excluding updated tuples) in the answer set of a CQ. The retransmission overhead is specified by the number of bits transmitted as the retransmission messages which consist of the updated tuples and their control messages. We capture the information about these two overheads in a parameter called *overhead ratio* that can be defined as follows:

Definition 4.4.1 *The overhead ratio V_i during the i^{th} evaluation period is the ratio of control message overhead C_i over retransmission overhead R_i during that period. It is specified by the formula*

$$V_i = \frac{C_i}{R_i}$$

APT uses the *overhead ratio* as a measure to evaluate the performance with w for the last evaluation period. Comparing the values of the *overhead ratios*

for the last two evaluation periods, APT decides how to adjust w for the next evaluation period. At the i^{th} evaluation, the window size is adjusted by using the following formula:

$$D_i = V_i - V_{i-1}$$

- $D_i > 0$ means that the control message overhead relative to the retransmission overhead during the i^{th} evaluation period is higher when compared to the $(i-1)^{th}$ evaluation period. So, the window size should be increased to reduce the control message overhead.
- $D_i < 0$ means that the retransmission overhead relative to the control message overhead during the i^{th} evaluation period is higher when compared to the $(i-1)^{th}$ evaluation period. So, the window size should be decreased to reduce the retransmission overhead.

Formally,

$$w = \begin{cases} w + \epsilon & \text{if } D_i > 0 \\ w - \epsilon & \text{if } D_i < 0 \\ w & \text{otherwise} \end{cases}$$

It can be easily confirmed that the probability that an updated tuple will be retransmitted depends on the value of w. Large values of w increase the retransmission probability while the small values of w decrease that probability. Similarly, the value of w also affects the availability of the tuples in the answer set in case of disconnections. Large values of w makes it possible for the MH to have more tuples compared to the case with small values of w.

4.5 Mixed Transmission (MT) Approach

APT presented above maintains a single window size for the whole database. This approach does suffer from the following shortcoming. The database may consist of a mixture of frequently changing objects (e.g., moving objects like cars) and rarely changing objects (e.g., motels). It may happen in this database

system that w cannot be increased because of the heavy retransmission overhead caused by frequently changing objects. On the other hand, small values of w are not appropriate for rarely changing objects since this would increase the control message overhead although this overhead is supposed to be minimal for such objects.

In order to handle the above problem, the MT approach partitions the database into two disjoint sets: one consisting of "hot" database objects (i.e., frequently changing) and the other consisting of "cold" database objects (i.e., rarely changing). This approach transmits the tuples referring the "cold" database objects as in the IT approach and the tuples referring the "hot" database objects as in the APT approach. Therefore, the control message overhead and the retransmission overhead is mostly limited to those associated with tuples referring to "hot" database objects. Consequently, we modify the definition of the overhead ratio to cover only "hot" database objects (O_h).

Definition 4.5.1 *The overhead ratio $V_i(O_h)$ for "hot" database objects during the i^{th} evaluation period is the ratio of control message overhead $C_i(O_h)$ over retransmission overhead $R_i(O_h)$ during that period. It is specified by the formula*

$$V_i(O_h) = \frac{C_i(O_h)}{R_i(O_h)}$$

MT decides how to adjust w for the next evaluation period using the following equation.

$$D_i(O_h) = V_i(O_h) - V_{i-1}(O_h)$$

Formally,

$$w(O_h) = \begin{cases} w(O_h) + \epsilon & \text{if } D_i(O_h) > 0 \\ w(O_h) - \epsilon & \text{if } D_i(O_h) < 0 \\ w(O_h) & \text{otherwise} \end{cases}$$

Thus, the control message overhead for the tuples referring to "cold" objects is minimized by making use of the fact that those objects are rarely updated. The retransmission and control message overheads for the tuples referring to "hot" objects is reduced by transmitting these tuples as in APT.

The availability of the tuples in the answer set of a CQ in case of disconnection can be considered separately for the tuples referring to "cold" and "hot" objects. All the tuples referring to "cold" objects will be available to the MH but the availability of the tuples referring to "hot" objects will depend on the current value of $w(O_h)$.

Chapter 5

Simulation Model

We have designed a simulation model to compare the performance of tuple transmission approaches IT, DT, PT, APT, and MT under different settings of environmental parameters such as the data update rate and disconnection period. Our simulation model is based on the performance models proposed in previous related works such as [BJ96, LS97]. These models have been extended to support modeling of processing location-dependent CQs.

Having more than one cell in the simulation model brings hand-offs into the picture. Suppose that an MH transmitted a query in the cell serviced by MSS_1 and moved to a new cell serviced by MSS_2 before the completion of the query. The only way MSS_1 can transmit the tuples in the answer set of the query is by sending tuples to MSS_2 over the fixed network so that MSS_2 can forward the tuples to the MH as long as the MH stays in its current cell.

Considering the existence of more than one cell in the simulation model introduces the communication overhead over the fixed network. Since the fixed network has a high bandwidth compared to the wireless network [1], we think that the mobility of MHs in multiple cells would not affect the relative performance of tuple transmission approaches in terms of communication overhead. Therefore, to eliminate the unnecessary details from our simulation model, we assume that the mobile system is limited to a single cell managed by a central

[1] ATM provides 155 Mbps and the current cellular technology provides bandwidth in the order of 10 Kbps.

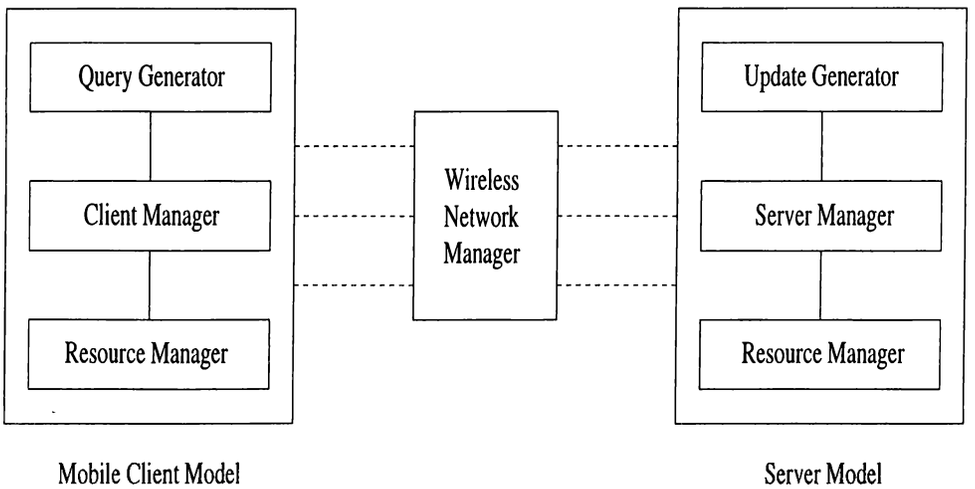

Figure 5.1: The Simulation Model.

data server (lying on an MSS) and a fixed number of mobile clients.

As shown in Figure 5.1, the simulation model consists of three basic components:

1. Mobile Client Model
2. Wireless Network Manager
3. Server Model

In the following sections, we describe each component in detail.

5.1 Mobile Client Model

Each mobile client is formed of 3 modules as shown in Figure 5.2: a Resource Manager which models the client CPU for handling the query results, a Query Generator which generates the query requests, and a Client Manager which processes the query requests and passes them to the server, models the disconnection operation, and receives and processes the tuples transmitted from the server.

Client queries are submitted from an MH to the server to be processed and

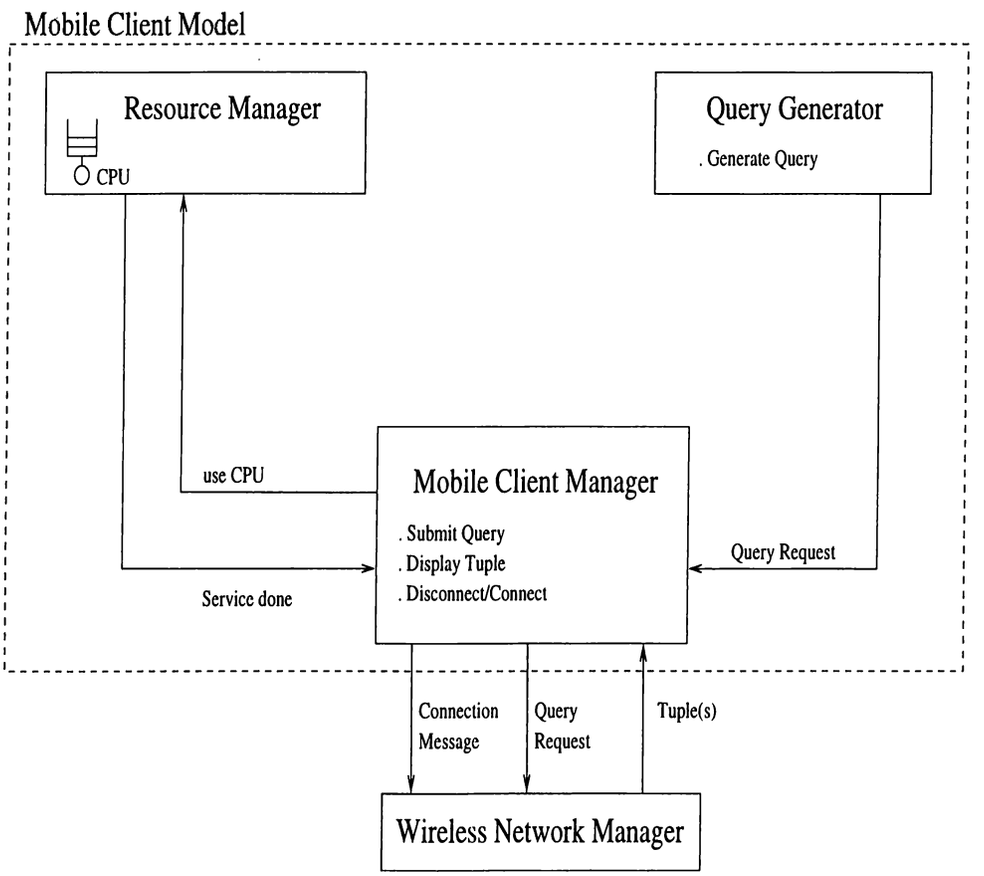

Figure 5.2: Mobile Client Model.

Parameter	Meaning
NumMobileHosts	Number of MHs
QueryRequestSize	Size of a CQ submitted by an MH
ThinkTime	Mean think time between queries in connect mode
DisconnectTime	Mean disconnect time
MaxQueryDuration	Maximum query duration
DisconnectProb	The probability that the MH will be disconnected after issuing a query
ClientMsgTime	CPU time to process a message per byte basis
ConnectMsgSize	Size of a connection indication message

Table 5.1: Mobile Client Model Parameters

a message (messages) containing the tuples that form the answer to the query is (are) transmitted back to the MH. The messages containing the tuples are processed by the MH and the tuples are displayed on the screen of the MH accordingly.

Table 5.1 lists the parameters of the Mobile Client Model. Each of the *NumMobileHosts* MHs generates a single stream of CQ with size *QueryRequestSize*. The arrival of a new query is separated from the completion of the previous query by an exponentially distributed think time with a mean of *ThinkTime*. The query duration is chosen randomly by the Query Generator and has the maximum value *MaxQueryDuration*. The probability that an MH will enter into a disconnection mode after issuing a query is determined by using *DisconnectProb* and the time delay before the disconnection is chosen uniformly within the execution time of the issued query. The duration that the MH will stay disconnected is chosen from an exponential distribution with a mean of *DisconnectTime*. When the MH later reconnects to the network, it sends a message having size *ConnectMsgSize* to inform the Server Manager.

No I/O time is modeled in the Resource Manager Module since we assume that the buffer pools of MHs are large enough to hold all the tuples received in response to an issued CQ. Each MH has a single CPU and the CPU time for processing a message per byte basis is determined by *ClientMsgTime*.

Parameter	Meaning
NetworkBandwidth	Wireless network bandwidth
ControlMsgSize	Size of a control message on the wireless network

Table 5.2: Wireless Network Manager Parameters

5.2 Wireless Network Manager

Table 5.2 lists the parameters of the Wireless Network Manager. The Wireless Network Manager component assumes that all messages are of equal priority that will be served on a First-Come First-Served (FCFS) basis with a service rate of *NetworkBandwidth*. When a message is to be transmitted, it is appended to a control message having size *ControlMsgSize*.

When the Wireless Network Manager finds out (i.e., while sending a message to an MH) that an MH is disconnected, it informs the Server Manager about the disconnection so that the transmission of the tuples to the MH can be paused until the MH reconnects to the network.

5.3 Server Model

The central server model has 3 modules as shown in Figure 5.3: a Resource Manager Module which models the server CPU time for query and update processing, an Update Generator which generates update requests, and a Server Manager Module which coordinates the query requests from MHs and update requests from the Update Generator.

The input parameters for the Server Model are listed in Table 5.3. The Resource Manager Module that models the database and physical resources of the system has *NumCPU* CPUs. The CPU time for processing a query and an update are specified by the parameters *ServerQueryTime* and *ServerUpdateTime*, respectively. All query and update requests are processed with the same priority on an FCFS basis. The database is modeled as a collection of *DatabaseSize* objects each with size *ObjectSize*. No I/O operation is modeled

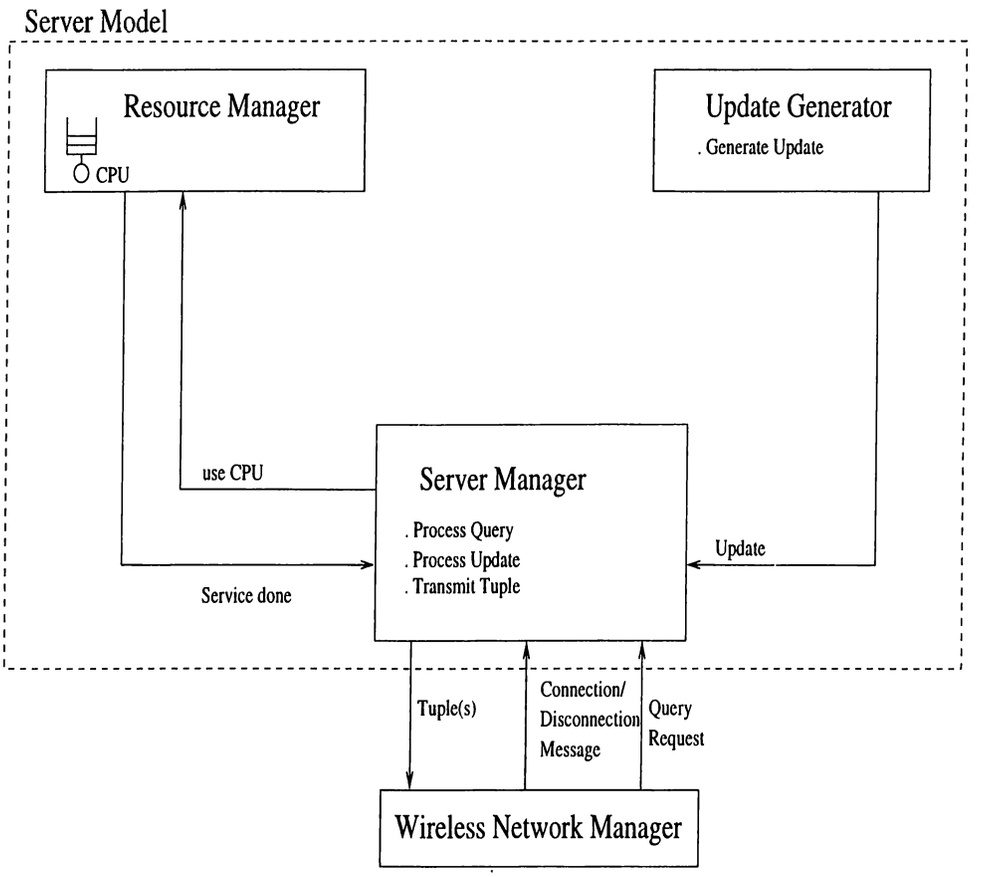

Figure 5.3: Server Model.

Parameter	Meaning
$NumCPU$	Number of CPUs
$ServerQueryTime$	Service time for a query in the data server
$ServerUpdateTime$	Service time for an update in the data server
$DatabaseSize$	Number of objects in the database
$ObjectSize$	Size of a database object
$QueryDuration$	Duration of the CQ issued by an MH
$MaxNumTuple$	Maximum number of tuples that can satisfy a CQ
$TupleSize$	Size of a tuple
$EvaluationPeriod$	Time period to adjust the window size
$WindowSize$	Initial window size
ϵ	Threshold value for the adjustment of the window size
$UpdateArrTime$	Mean interarrival time between updates
$HotUpdateBounds$	Data object bounds of hot update range
$ColdUpdateBounds$	Data object bounds of cold update range
$HotUpdateProb$	Probability that an update will be applied to a "hot" object
$HotQueryProb$	Probability that a tuple will refer to a "hot" object
$HotRemoveProb$	Probability that an updated tuple referring to a "hot" object will be removed from the corresponding answer set
$ColdRemoveProb$	Probability that an updated tuple referring to a "cold" object will be removed from the corresponding answer set

Table 5.3: Server Model Parameters

since we assume that the buffer pool in the server is large enough to hold the entire database.

Duration of a CQ submitted by an MH is determined by the MH and specified by the parameter *QueryDuration*. When a CQ is issued by an MH, it is processed by the Server Manager and the set of tuples satisfying the query are determined. The number of tuples in the answer set of a CQ is uniformly determined with a maximum of *MaxNumTuple* tuples. The size of each tuple is specified by *TupleSize*. If a query is executed by the server at time t, the *begin* time of a tuple in the answer set is uniformly distributed within the interval $[t, t + QueryDuration]$. Similarly, the *end* time for that tuple in the answer set is uniformly distributed within the interval $[begin, t + QueryDuration]$.

The Server Manager also decides when and which tuples should be transmitted to the MH depending on the underlying tuple transmission approach (i.e., one of the IT, DT, PT, APT, MT approaches). The window size is also adjusted by the Server Manager for the APT, and the MT approaches. The window size is evaluated and adjusted every *EvaluationPeriod* time units. Depending on the underlying policy, the window size is incremented or decremented by a small integer ϵ. We assume that the time needed to evaluate and adjust the window size is negligible and therefore do not take it into account in our model.

At the server, a single stream of updates is generated. These updates are separated by an exponentially distributed update interarrival time with a mean of *UpdateArrTime*. Our model can specify different update and query patterns. For the central data server, *HotUpdateBounds* and *ColdUpdateBounds* parameters are used to specify the "hot" and "cold" regions of the database respectively for update requests. *HotUpdateProb* and *HotQueryProb* specify the probability that an update will be applied to a database object in the "hot" database region and a tuple in the answer set of a CQ will refer to a "hot" object, respectively. *HotRemoveProb* and *ColdRemoveProb* specify the probability that an tuple referring to a "hot" object and a "cold" object will be removed from the answer set, respectively.

When a database object is explicitly updated, we assume that all the tuples

in the answer set of every CQ that refer to the updated object, are changed. For simplicity we ignore the possibility that the updated object may satisfy new queries that it did not satisfy before. We also assume that the attributes representing the position of the MH that issued the query do not change until the query processing is completed; because, such a change results in the reevaluation of the query and in this study we focus on the retransmissions rather than the reevaluations. However, this assumption does not mean that the querying MH is a stationary object.

When a tuple in an answer set is updated, it is immediately retransmitted to the corresponding MH. The original tuple (before update) may be in use at the MH at the time of the update and MH must be informed about the update to the tuple immediately so that it can invalidate the original tuple.

When the Wireless Network Manager detects that an MH is disconnected, it informs the Server Manager to pause transmitting tuples to the MH until it reconnects to the network. When the MH reconnects, the Server Manager resumes transmitting the valid tuples (tuples with *end* time $\leq$ current time) to the MH.

Chapter 6

Experiments and Results

In this chapter, we present the performance results for the tuple transmission approaches for CQs that we discussed in Chapter 4. A number of simulation experiments have been conducted to study the behavior of different tuple transmission algorithms under various data update rates, maximum query duration, disconnection period and update/query patterns.

Experiments were designed to evaluate the relative performance of the algorithms in terms of communication overhead imposed on the wireless network and the availability of tuples in case of disconnections. All experiments were performed on SunSparc Workstations running SUNOS, using the CSIM [Sch92] simulation package. Each experiment was run until a total of 5000 CQs are completed. Each experiment is repeated 30 times with different *seeds* in order to obtain a statistically significant sample of CQs. The presentation of performance results is preceded by a discussion of the performance metrics and the parameter settings.

6.1 System Performance Metrics

The primary performance metric in this study is the average number of bits transmitted to an MH in response to a CQ. The number of bits transmitted for a CQ is computed by summing up the total number of bits transmitted

as tuples and control messages in response to a CQ. Another metric used is the *availability* of tuples in the answer set of a CQ in case of a disconnection. The availability of tuples in case of a disconnection is specified as the ratio of the number of tuples received by the MH prior to disconnection over the total number of tuples that would have been received by the end of the disconnection period if the MH had been connected to the network.

6.2 Parameter Settings

The values of the simulation parameters were chosen so as to be comparable to the related simulation studies such as [BJ96, LS97]. Since there is no data available for modeling the tuples in the answer set of a CQ, we are concerned here with performance trends rather than with exact performance predictions.

Table 6.1 provides the values of the simulation parameters which are common to all experiments except where otherwise specified. There are 100 MHs and the mean think time between queries for an MH is 1000 seconds. The maximum duration of a query an MH can request is varied from 240 seconds to 360 seconds in order to examine how query duration affects the performance of the tuple transmission approaches. The size of a CQ request is 256 bytes. An MH disconnects from the network after it issues a CQ once per 10 queries. The mean disconnection time is varied from 50 to 1000 seconds in order to observe the performance trends of the tuple transmission approaches in case of both short and long disconnections. When the MH reconnects to the network after the disconnection period, it sends a 4 byte message to the Server indicating the reconnection. The CPU time for processing a byte while sending/receiving messages is 0.0001 second.

The bandwidth of the wireless network is 19200 bits per second which is a reasonable data transmission rate in current cellular network technology. Each message to be transmitted is appended to a 256 byte control message by the Wireless Network Manager.

The database is modeled to be consisting of 1000 database objects with an

Parameter	**Value**
$NumMobileHosts$	100
$ThinkTime$	1000 s
$MaxQueryDuration$	varied from 240 s to 360 s
$QueryRequestSize$	256 bytes
$DisconnectProb$	1/10
$DisconnectTime$	varied from 50 s to 1000 s
$ConnectMessageSize$	4 bytes
$ClientMsgTime$	0.0001 s/byte
$NetworkBandwidth$	19200 bps
$ControlMsgSize$	256 bytes
$DatabaseSize$	1000 objects
$ObjectSize$	256 bytes
$TupleSize$	264 bytes
$UpdateArrTime$	varied from 1 s to 5 s
$HotUpdateBounds$	All the database
$NumCPU$	1
$ServerQueryTime$	0.01 s
$ServerUpdateTime$	0.02 s
$MaxNumTuple$	40 tuples
$HotRemoveProb$	0.01
$WindowSize$	180 s
$EvaluationPeriod$	500 s
ϵ	1 s

Table 6.1: Parameter Settings for The Base Experiment

object size of 256 bytes. A tuple contains a database object plus 4 bytes for each of the *begin* and the *end* attributes. The interarrival time of the database updates is varied from 1 second to 5 seconds in order to observe the behavior of the tuple transmission approaches under various levels of update rates. Unless otherwise specified, it is assumed that all the database consists of "hot" objects. The server has a single CPU and the server CPU times for processing a query and an update are set to 0.01 seconds and 0.02 seconds, respectively.

An answer to a CQ can contain at most 40 tuples. The probability that an updated tuple will be removed from the corresponding answer set is set at 0.01. The initial window size for the PT, and the APT approach is 180 seconds which was experimentally observed to provide the best performance. The window size is evaluated every 500 seconds and can be incremented or decremented by 1 second.

6.3 The Base Experiment

We first examine the performance results of the proposed tuple transmission approaches under varying data update rates by setting $MaxQueryDuration$ and $DisconnectTime$ to 300 seconds. Performance of the MT approach is not examined in this experiment because the behavior of MT is the same as that of APT since all the database objects are assumed to be "hot". Figures 6.1 through 6.4 show the performance results obtained.

As illustrated in Figure 6.1, DT performs the worst among all tuple transmission approaches in terms of the average number of bits transmitted in response to a CQ. This result is due to involving the highest control message overhead caused by the transmission of each tuple separately as shown in Figure 6.2. At low data update rates the performance results of IT, PT, and APT are close to each other. Transmitting all the tuples at once or transmitting them periodically with $w = 180$ seconds in PT and APT, does not make much difference in terms of the control message overhead. As Figure 6.2 shows, the control message overhead involved with IT is close to that of PT and APT at low data update rates.

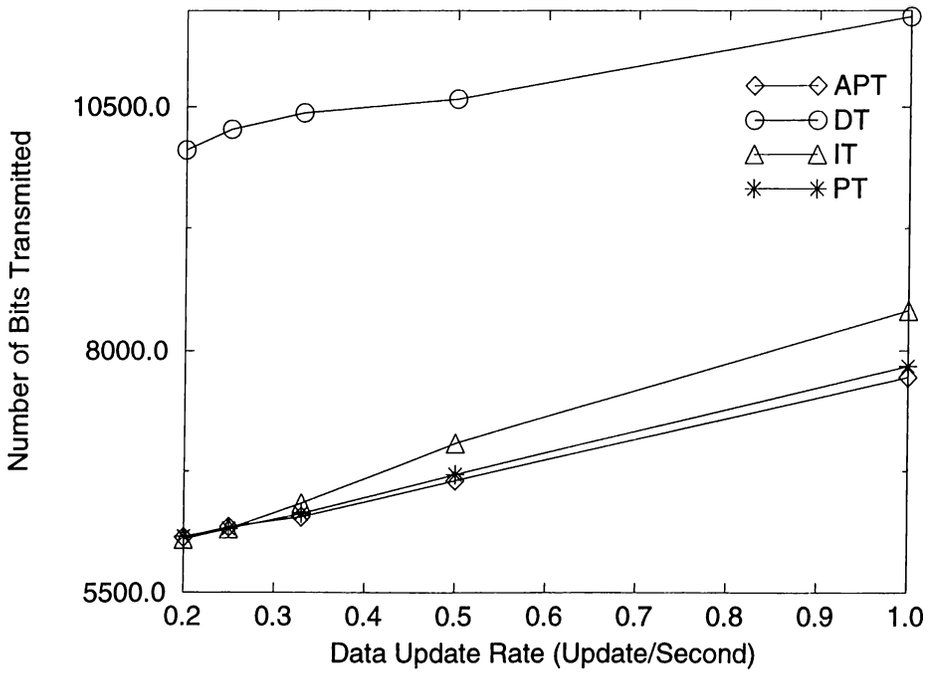

Figure 6.1: Average Number of Bits Transmitted vs Data Update Rate.

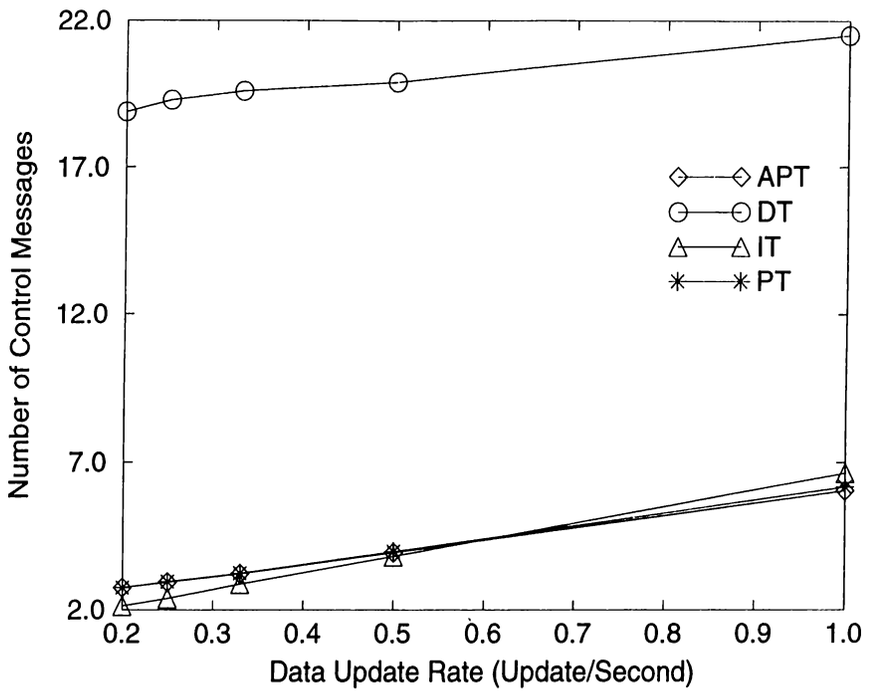

Figure 6.2: Average Number of Control Messages vs Data Update Rate.

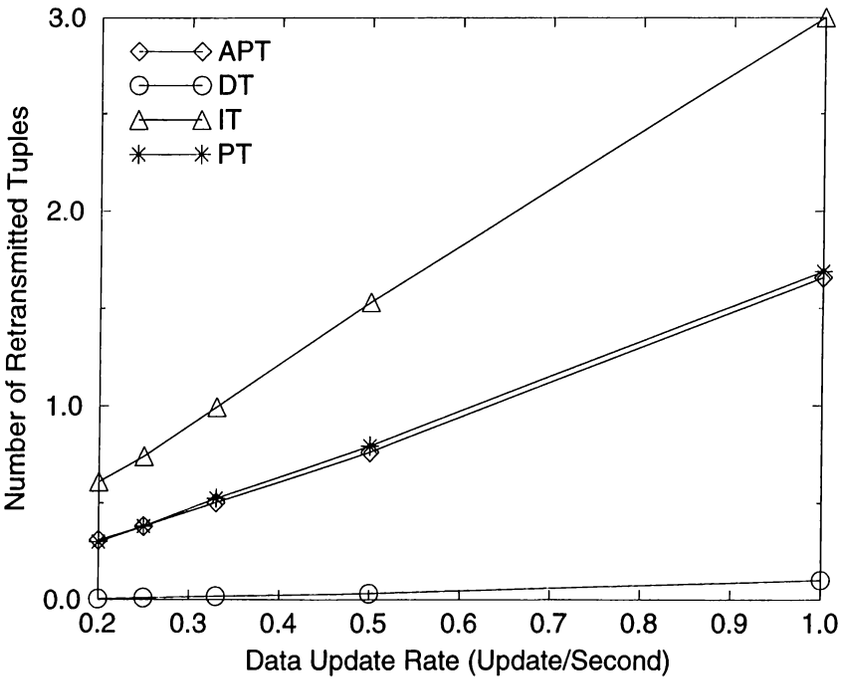

Figure 6.3: Average Number of Retransmitted Tuples per CQ vs Data Update Rate.

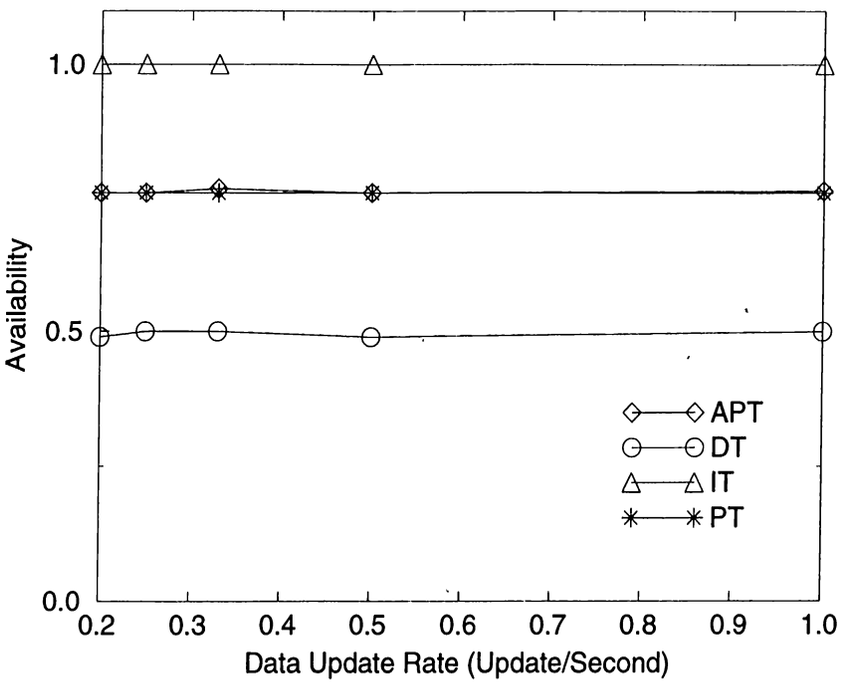

Figure 6.4: Availability of Tuples vs Data Update Rate.

As the data update rate is increased, all the curves start to move upward due to the increasing retransmission overhead as shown in Figure 6.3. Furthermore, the performance difference between IT, PT, and APT in terms of the average number of bits transmitted becomes apparent with the high data update rates. PT and APT approaches have an important benefit over IT in terms of the retransmission overhead. Another observation is that the periodic adjustment of w according to the criterion we have formulated in APT approach provides some improvement over the performance of PT.

The reader may notice from Figure 6.3 that the number of retransmissions per CQ may not always be zero with the DT approach. This may seem contradictory as we have limited the scope of retransmissions to those of Case 2 (in Chapter 3) which exclude the retransmissions due to an update after the *begin* time of a tuple. However, when a tuple is changed due to an explicit update to the database, it is immediately retransmitted. Therefore, Case 2 retransmissions are also possible with DT.

As we discussed before, supporting the ability for an MH to work in the stand-alone mode in case of disconnections can be very important in some applications. Figure 6.4 shows the availability of tuples in the answer set of a CQ in case of disconnections. As expected, IT has the highest availability since this approach transmits all the tuples together as soon as they are determined. The performances of PT and APT in terms of availability are nearly the same. DT is the worst approach in supporting the stand-alone working ability since the transmission of a tuple is delayed until its *begin* time. We also observe that increasing data update rate does not have an impact on the performance of any approach in terms of availability.

6.3.1 Evaluation of the Impact of Query Duration

In this experiment, we examine the performance in terms of the average number of bits in response to a CQ for the four tuple transmission approaches as the maximum query duration is varied while setting $UpdateArrTime$ to 1. Increasing the maximum query duration increases the probability that a tuple will be updated therefore the probability that it will be retransmitted as shown

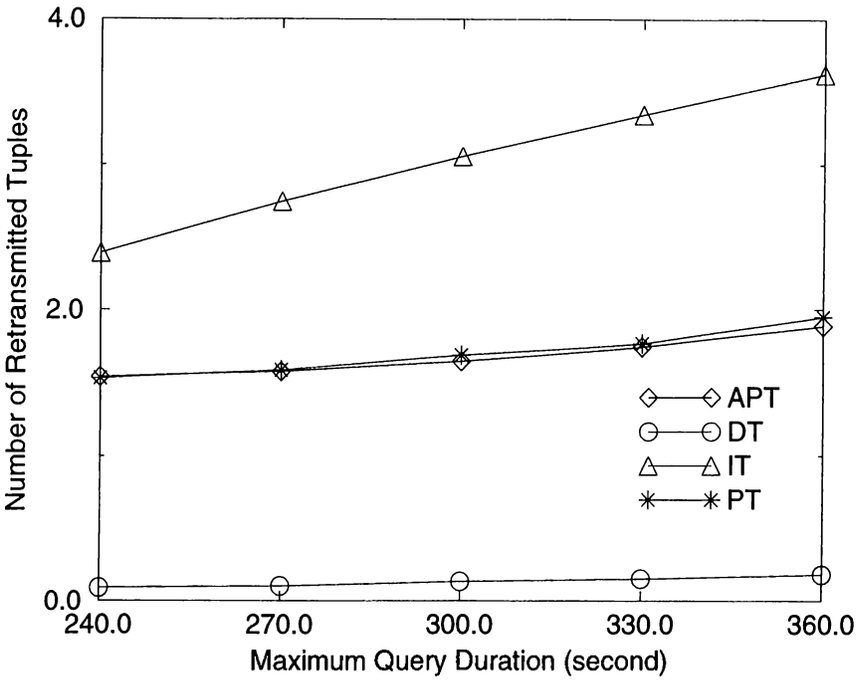

Figure 6.5: Average Number of Retransmitted Tuples vs Maximum Query Duration.

in Figure 6.5. This increase is dramatic in IT since IT is the most prone approach to retransmissions.

The average number of bits transmitted in response to an issued CQ increases as the query duration increases as shown in Figure 6.6. As compared to Figure 6.1, the relative performance of the approaches does not change and DT is still the worst performing approach. The performance difference between IT and the other two approaches PT and APT becomes more apparent as the query duration is increased.

6.3.2 Evaluation of the Impact of Disconnection Period

In this section, we investigate the impact of the time period an MH stays disconnected after issuing a CQ on the availability of the tuples. $MaxQueryDuration$ is set to 300 seconds. As expected, the availability of tuples in IT is always 1 independent of the disconnection period. The availability of tuples in all the

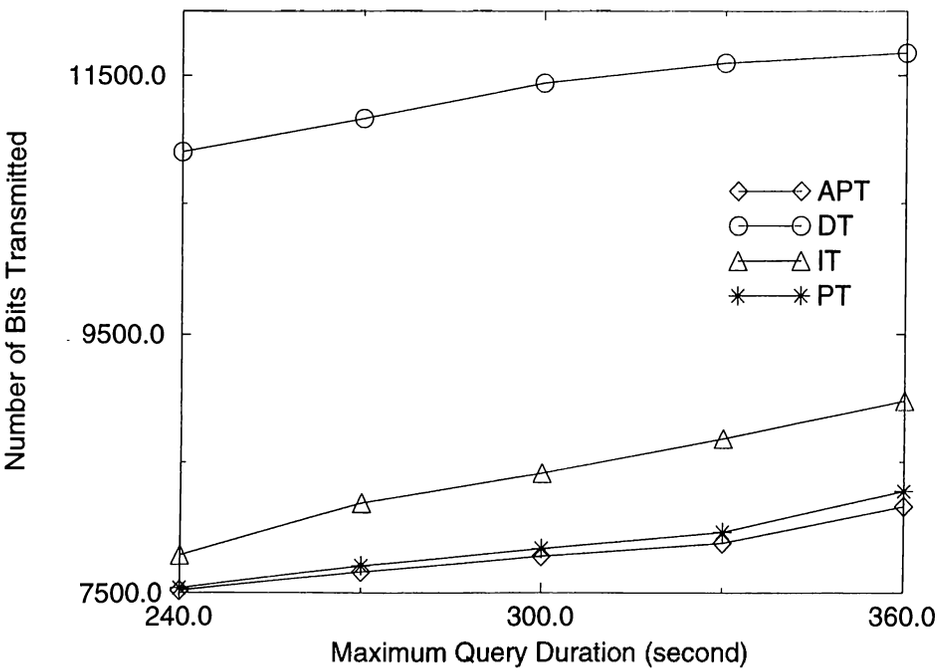

Figure 6.6: Average Number of Bits Transmitted vs Maximum Query Duration.

other approaches decreases up to a certain point as the disconnection period increases as shown in Figure 6.7. After that particular point, the availability of tuples remains constant. This is a reasonable result because the availability of tuples is the same for disconnection periods longer than the query duration. Suppose that an MH issued a CQ with duration 200 seconds and is disconnected. The availability of tuples will be the same for disconnections lasting more than 200 seconds.

6.4 Evaluation of the Impact of Hotspots

As we discussed earlier the database may consist of a mixture of frequently changing and rarely changing objects. We set up an experiment in order to observe the performance of five tuple transmission approaches in case there exists a hotspot in the database. Table 6.2 lists the values of the related parameters used in this experiment. According to the new update pattern in

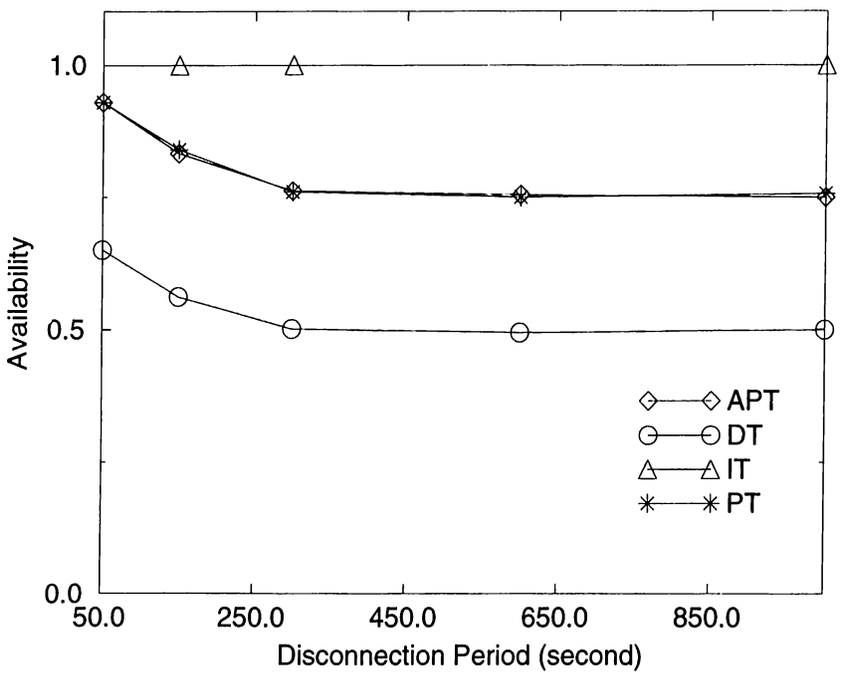

Figure 6.7: Availability of Tuples vs Disconnection Period.

this experiment, 80% of the updates are applied to 20% of the database and the object a tuple will refer to is uniformly chosen from the database.

The comparison of Figures 6.1 and 6.8 shows that the relative performance of IT, PT, and APT does not change in terms of the average number of bits transmitted in response to a CQ. MT performs the best in terms of reducing the communication overhead. Figures 6.9 and 6.10 show that transmitting

Parameter	**Value**
$HotUpdateBounds$	1-200
$ColdUpdateBounds$	201-1000
$HotRemoveProb$	0.01
$ColdRemoveProb$	0.04
$HotUpdateProb$	0.8
$HotQueryProb$	0.2
$WindowSize$	180 s (for PT & and APT), 150 s (for MT)

Table 6.2: Parameter Settings

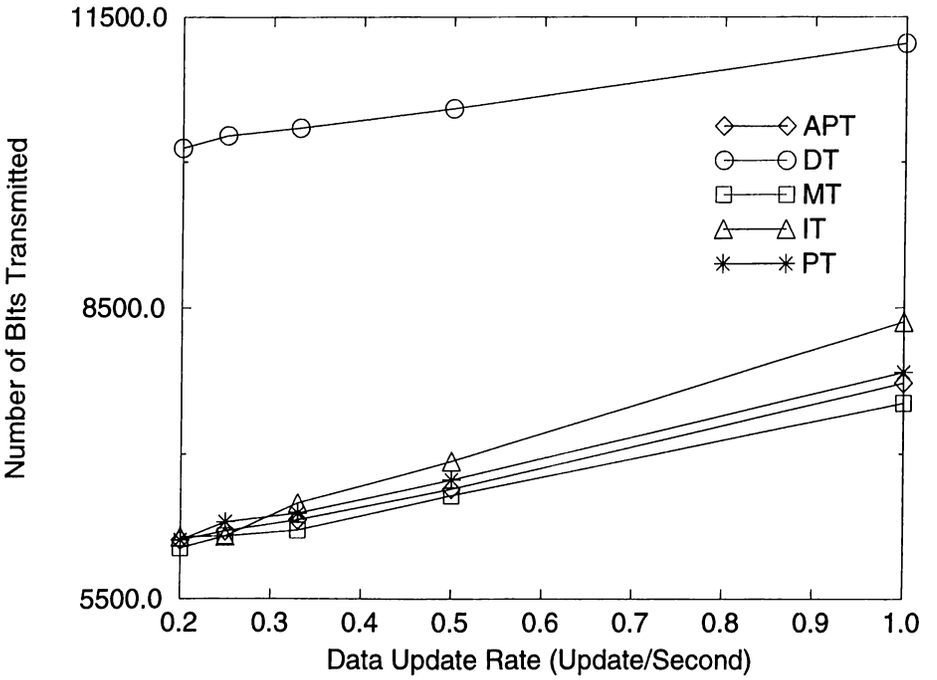

Figure 6.8: Average Number of Bits Transmitted vs Data Update Rate.

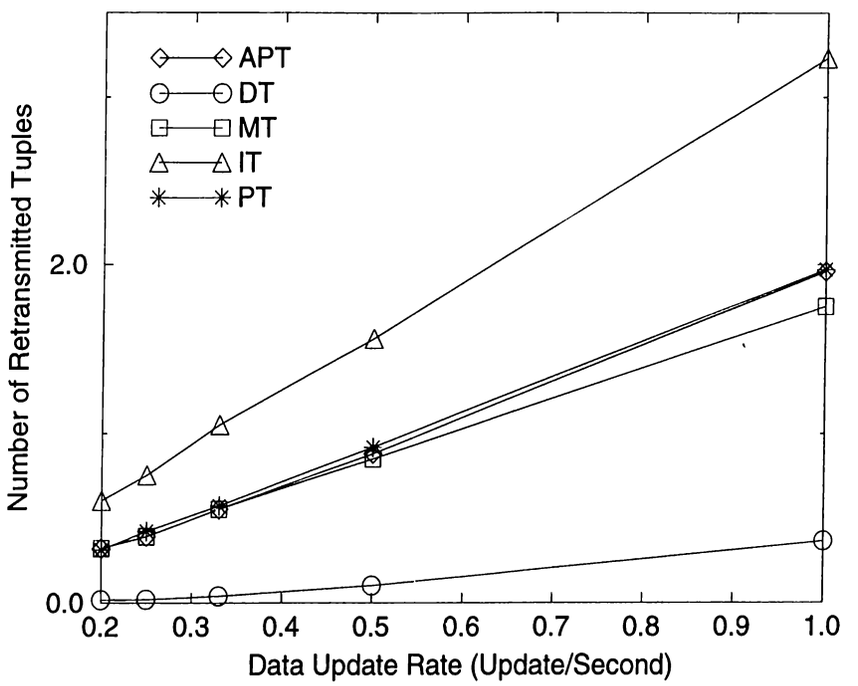

Figure 6.9: Average Number of Retransmitted Tuples vs Data Update Rate.

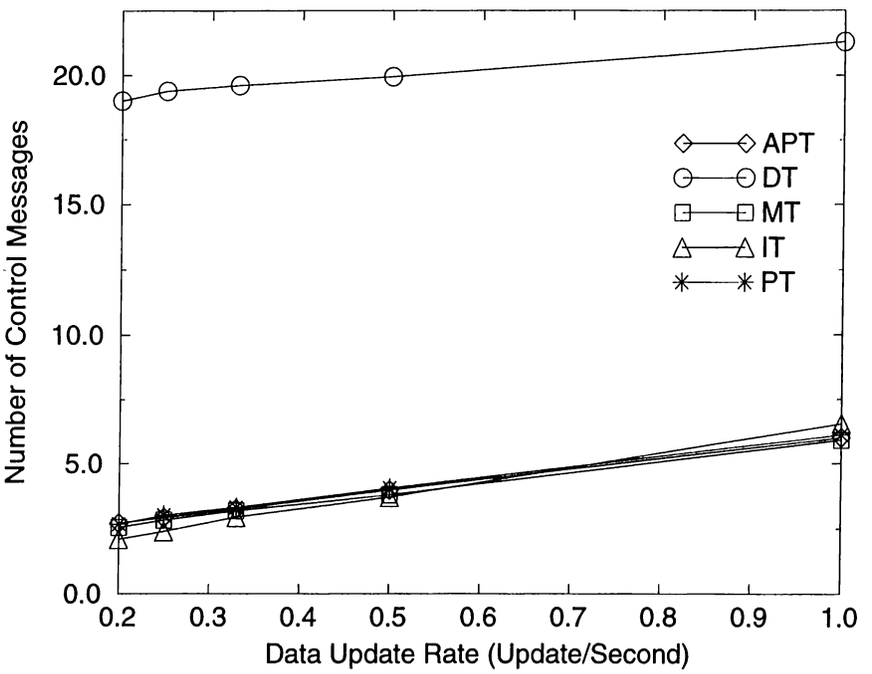

Figure 6.10: Average Number of Control Messages vs Data Update Rate.

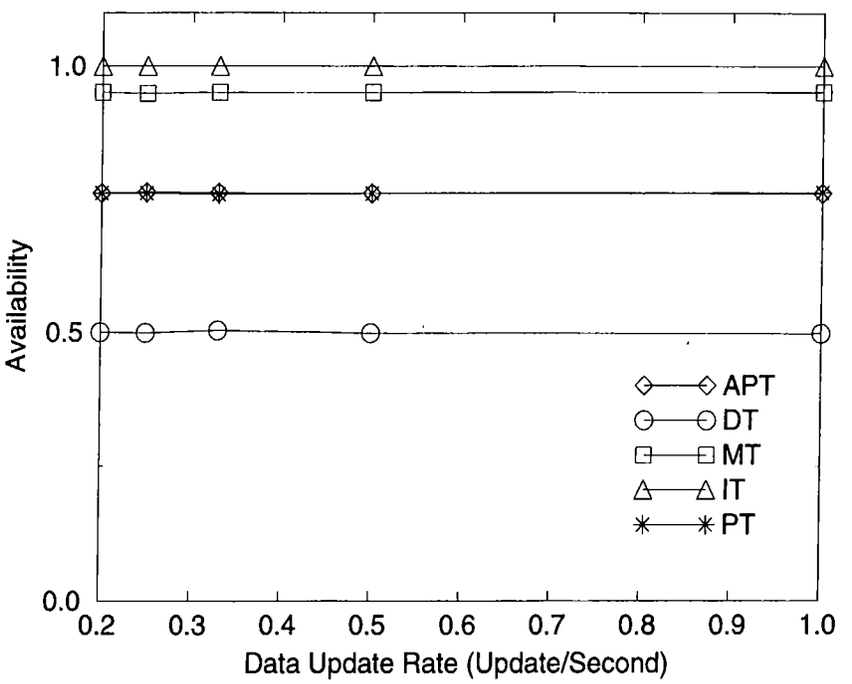

Figure 6.11: Availabiiity of Tuples vs Data Update Rate.

the tuples referring to "hot" objects as in APT and the tuples referring to "cold" objects as in IT reduces the communication overhead and offers the best performance among all the tuple transmission approaches we presented in case of a hotspot in the database.

MT performs closer to the performance of IT than that of DT, PT, and APT in terms of the availability of tuples in case of a disconnection (Figure 6.11). Since the tuples forming the answer set are chosen uniformly from the database, 80% of the tuples in the answer set of a CQ are supposed to be those referring to "cold" objects. MT transmits the tuples referring to "cold" objects immediately at the time they are determined. Therefore, those tuples are always available to the MH in case of a disconnection. Tuples referring to "hot" objects are partially available in case of a disconnection and the above combination leads to a higher availability of tuples than those of PT and APT.

6.4.1 Evaluation of the Impact of Query Duration

In this experiment, we examine the performance in terms of the average number of bits in response to a CQ for the five tuple transmission approaches as the maximum query duration is changed. Figure 6.12 shows the performance results. Existence of a hotspot in the database does not change the overall effect of increasing query duration. The results shown in Figure 6.12 are close to that of Figure 6.6. Additionally, MT still performs the best as the maximum query duration is changed.

6.4.2 Evaluation of the Impact of Disconnection Period

We investigate the impact of the time period an MH stays disconnected after issuing a CQ on the availability of the tuples. Figure 6.13 shows the performance results. Our observations with the experiment of Section 6.3.2 are also valid in this experiment. MT provides a performance between IT and the remaining tuple transmission approaches in terms of availability of tuples in case of a disconnection. The performance of MT at the point after which the

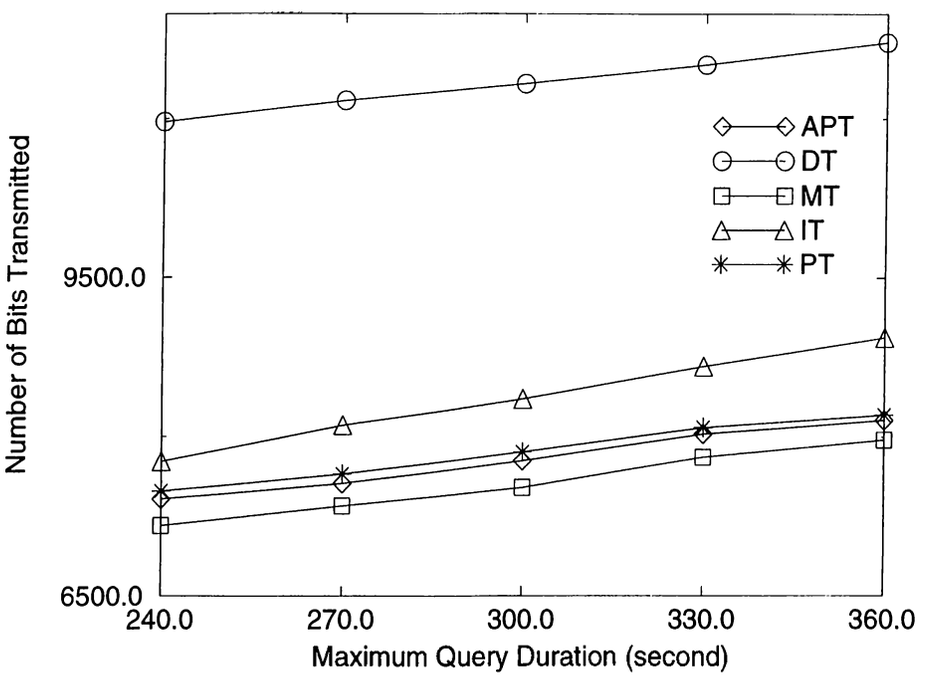

Figure 6.12: Average Number of Bits Transmitted vs Maximum Query Duration.

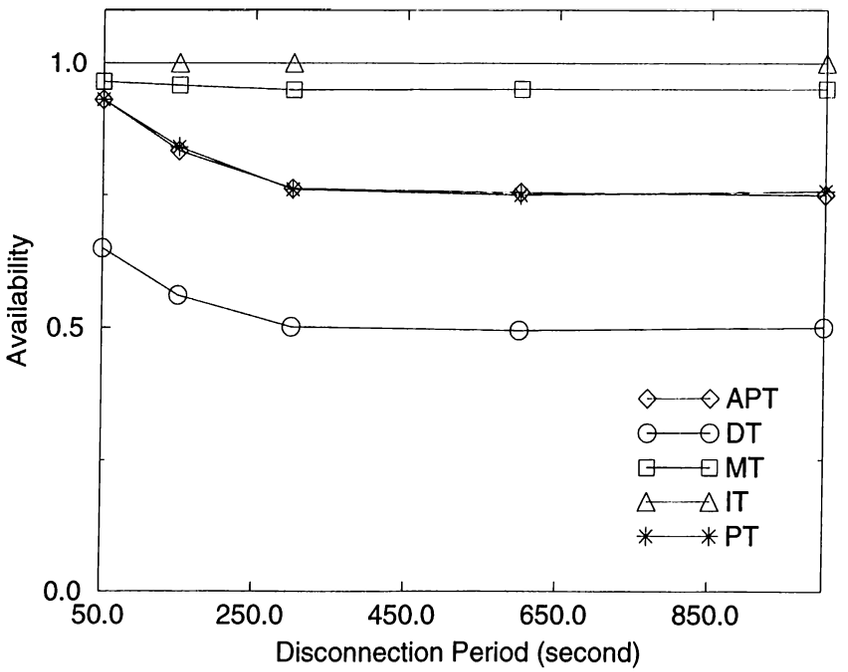

Figure 6.13: Availability of Tuples vs Disconnection Period.

availability of tuples remains constant is consistently higher compared to the performance of DT, PT, and APT.

6.4.3 Evaluation of the Impact of Query Hotspots

In a mobile database system, it is also possible that some objects are accessed more frequently than the others. We examine the impact of such a situation in this experiment by setting *HotQueryProb* to 0.8. It means that the hotspot of updates which is bounded by *HotUpdateBounds* is now a hotspot in terms of access workload too.

As shown in Figure 6.14, the new query pattern does not change the relative performance of the tuple transmission approaches we propose in terms of the average number of bits transmitted in response to a CQ. However, the performance difference between MT and the other approaches becomes more apparent with the high data update rates. In this experiment, the answer to

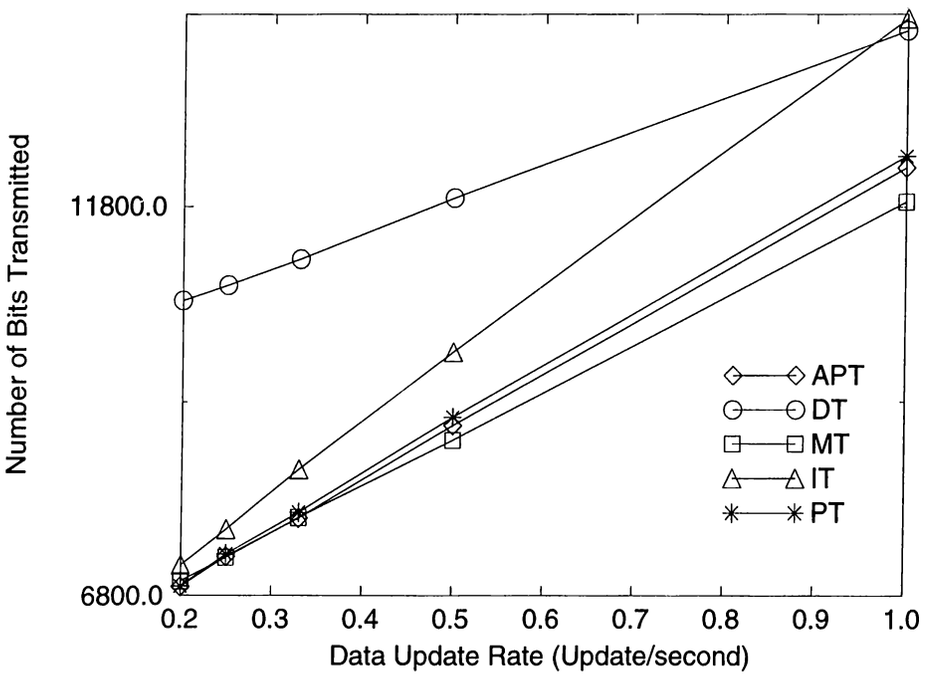

Figure 6.14: Average Number of Bits Transmitted vs Data Update Rate.

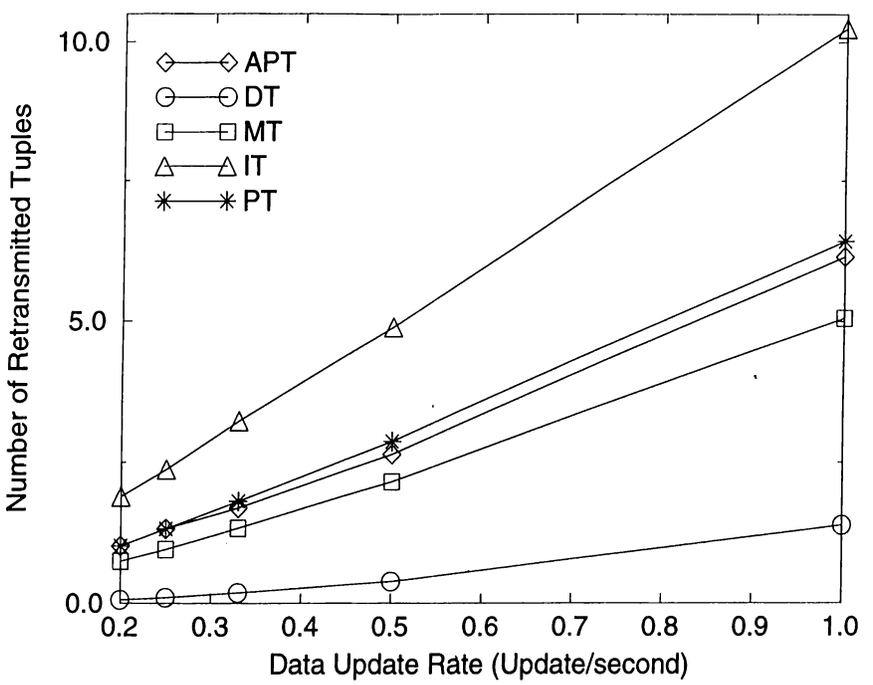

Figure 6.15: Average Number of Retransmitted Tuples per CQ vs Data Update Rate.

a CQ consists of mostly "hot" objects. Therefore, retransmissions are more frequent compared to the previous experiments (see Figure 6.15). Heavy retransmission overhead makes IT perform worse than DT with an update rate of 1 update per second.

Chapter 7

Conclusions and Future Work

In a mobile computing environment, a user (a mobile host) with a wireless connection to the information network is not required to maintain a fixed position in the network. The characteristics of mobile computing environments such as frequent disconnections of mobile hosts (MHs), significant limitations of bandwidth and power, resource restrictions, and fast-changing locations make the existing hardware and software systems inadequate to be used in such environments. Therefore, traditional networking and data management techniques need to be improved based on the requirements of mobile computing environments.

MHs can access information via submitting queries to the information server over the network. Some of these queries are called *location-dependent queries* as the answer to such kind of queries depends on the current location of the user who issued the query [SWCD97, SWCD98, TUW98, WXCJ98]. As an example, consider the following query: "display motels that are within 5 miles of my position" submitted to a hotel management system by a person driving a car. Suppose that the above query is submitted as a *continuous query* (CQ) [SWCD97, SWCD98] (i.e., answer to the query has to be continuously updated as the car moves). It is clear that the answer to the query changes with the car movement and continuously transmitting a new answer to the MH depending on the new position of the car would impose a serious performance overhead.

Moving Objects Spatio-Temporal (MOST) data model proposed in [SWCD97, SWCD98] returns a set of tuples as the answer to a CQ. The answer set consists of tuples $< S, begin, end >$ indicating that object S is the answer of the CQ from time *begin* to time *end*. Once the answer to a CQ is determined, an interesting question arises in MOST: when should the tuples in the answer set be transmitted?

The answer to the above question is critical in the sense that it can affect the communication overhead imposed on the network, the power consumption at MHs and the availability of tuples in case of disconnections. For example, the *Immediate Transmission* (IT) approach [SWCD97, SWCD98] which transmits all the tuples to the MH at the time they are determined causes too much retransmission overhead due to the updates to the database, whereas the *Delayed Transmission* (DT) approach [SWCD97, SWCD98] which delays the transmission of a tuple until its *end* time causes each tuple to be transmitted in a separate message and therefore too much control message overhead.

In our thesis, we present three new approaches for the transmission of the tuples in the answer set of a location-dependent CQ. The first approach called *Periodic Transmission* (PT) transmits the tuples in the answer set periodically. At each w time units, this method transmits all the tuples $< S, begin, end >$ satisfying the condition $t \leq begin < t + w$ where t is the current time and w is the size of the time window. In the second approach which we call *Adaptive Periodic Transmission* (APT), as an extension to the first approach, w is dynamically adjusted according to the communication overhead changing due to environmental parameters such as data update rate, disconnection frequency, and disconnection period. The final approach, called *Mixed Transmission* (MT), differs from the first two approaches in that data objects are partitioned into two groups: one consisting of "hot" objects of updates and the other of "cold" objects of updates. This approach transmits the "hot" tuples as in APT and "cold" tuples as in IT.

We have also designed and implemented a simulation model that supports modeling of processing location-dependent CQs in order to examine the presented tuple transmission approaches under different setting of environmental parameters. In particular, we have examined the effects of alternative tuple

transmission approaches in terms of the average number of bits transmitted in response to a CQ and the availability of tuples in the answer set of a CQ in case of a disconnection while varying data update rate, maximum query duration, disconnection period and update/query pattern.

The experiments conducted demonstrate that the control message overhead caused by DT dominates the retransmission overhead caused by IT. In other words, DT performs worse than IT in terms of the average number of bits transmitted in response to a CQ. Furthermore, PT reduces the retransmission overhead of IT and the control message overhead of DT and therefore leads to a better performance. We also observed that by adjusting the window size on the basis of the communication overhead, APT offers a small improvement over PT. The final observation is that, relying on the existence of hotspots in the database, MT performs the best among all the approaches presented in this study.

As far as the availability of tuples is concerned, IT performs the best in all the experiments conducted. The performance of MT which is a combination of IT and APT is worse than that of IT but better than that of APT. PT and APT perform close to each other. Finally, DT has the poorest performance in terms of supporting the stand-alone mode working capability of an MH.

We conclude that the choice between IT, MT, and APT depends on the answer to the following question: How critical is it for the MHs to be able to continue displaying tuples on the screen in case of disconnections? Once the tradeoff between the high availability of tuples in case of disconnections and the low communication overhead is determined, the appropriate tuple transmission approach would be one of the approaches we propose (MT, APT) for low communication overhead, or IT otherwise.

One possible direction of future research is to extend our simulation model to support caching of database objects at MHs. Frequently accessed database objects can be broadcast by the data server to MHs and stored in caches of MHs. In such a system, the data server processes the CQs submitted by MHs and transmissions will be limited to only those tuples referring to objects that are not cached at the MH. For the tuples referring to objects that are cached at

the MH, only *begin* and *end* attributes, and the object id need to be transmitted to the MH. Such a caching mechanism can reduce the communication overhead significantly especially if database objects are large. Integrating a caching strategy pops up new issues such as cache invalidation mechanisms and the period of the broadcast cycle which are subject to further investigation.

Another interesting direction of future research can be the investigation of distributed query processing issues in which the database might be distributed over MHs as well as fixed ones.

Bibliography

[AG93] Rafael Alonso and S. Ganguly. Query optimization for energy efficiency in mobile environments. In *Proceedings of the International Workshop on Foundations of Models and Languages for Data and Objects*, Aigen, Austria, 1993.

[AP95] B. Awerbuch and D. Pelig. Online tracking of mobile users. *Journal of the ACM*, 42(5):1021–1058, September 1995.

[BI93] D. Barbara and Tomasz Imielinski. Adaptive stateless caching in mobile environments: An example. Technical report, Matsushita Information Technology Laboratory, 1993.

[BI94] D. Barbara and Tomasz Imielinski. Sleepers and workaholics: Caching strategies for mobile clients. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1–12, 1994.

[BJ96] Omran Bukhres and Jin Jing. Analysis of adaptive caching algorithms in mobile environments. *Information Sciences*, pages 1–27, 1996.

[BMM96] Omran Bukhres, S. Morton, and M. Mosman. Mobile computing architecture for a battlefield environment. In *Proceedings of the International Symposium on Cooperative Database Systems for Advanced Applications*, 1996.

[Chr93] P. K. Chrysanthis. Transaction processing in mobile computing environment. In *Proceedings of IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 77–83, Princeton, New Jersey, October 1993.

[DHB97] M. H. Dunham, A. Helal, and S. Balakrishnan. A mobile transaction model that captures both data and movement behavior. *MONET*, 2(2):115–127, 1997.

[EJB95] Ahmed Elmagarmid, Jin Jing, and Omran Bukhres. An efficient and reliable reservation algorithm for mobile transactions. In *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95)*, 1995.

[FZ94] G. H. Forman and J. Zahorjen. The challenges of mobile computing. *IEEE Computer*, 27(6), April 1994.

[GA93] Sumit Ganguly and Rafael Alonso. Query optimization in mobile environments. Technical report, Rutgers University, December 1993.

[IB93] Tomasz Imielinski and B. R. Badrinath. Data management for mobile computing. *ACM SIGMOD RECORD*, 22(1):34–39, 1993.

[IB94] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Communication of ACM*, 37(10):18–28, October 1994.

[LS97] H. V. Leong and A. Si. Database caching over the air storage. *The Computer Journal*, 40(7), 1997.

[PB93] E. Pitoura and B. Bhargava. Dealing with mobility: Issues and research challenges. Technical Report TR-97-070, Department of Computer Sciences, Purdue University, 1993.

[PB94] E. Pitoura and B. Bhargava. Building information system for mobile environments. In *Proceedings of the 3rd International Conference on Information and Knowledge Management*, pages 371–378, Guithesburg, MD, November 1994.

[PS97] R. Prakash and M. Singhal. Dynamic hashing + quorum = efficient location management for mobile computing systems. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, page 291, Santa Barbara, August 1997.

[RB95] S. Rajagopalan and B. R. Badrinath. An adaptive location management strategy for mobile ip. In *Proceedings of the 1st ACM Mobicom*, November 1995.

[Sch92] H. Schwetman. *CSIM User's Guide.* MCC Corporation, 1992.

[SW98] A. P. Sistla and O. Wolfson. Minimization of communication cost through caching in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, 9(4):378–390, April 1998.

[SWCD97] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceedings of the 13th International Conference on Data Engineering*, pages 422–432, Birmingham, UK, April 1997.

[SWCD98] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. *Querying the Uncertain Position of Moving Objects*, chapter Temporal Databases: Research and Practice, pages 310–337. Lecture Notes in Computer Science (Springer Verlag), 1998.

[SWD+96] A. P. Sistla, O. Wolfson, S. Dao, K. Narayonan, and R. Raj. An architecture for consumer-oriented online database services. In *Proceedings of the 6th International Workshop on Research Issues in Data Engineering: Interoperability of Nontraditional Database Systems*, New Orleans, LA, February 1996.

[TKN96] M. Tsukamoto, R. Kadobayashi, and S. Nishio. Strategies for query processing in mobile computing. In *Mobile Computing*, pages 595–620. Kluwer Academic Publishers, 1996.

[TUW] Jamel Tayeb, Ozgur Ulusoy, and O. Wolfson. Indexing mobile objects for location-dependent queries. In preparation.

[TUW98] Jamel Tayeb, Ozgur Ulusoy, and O. Wolfson. A quadtree based dynamic attribute indexing method. *The Computer Journal*, 41(3), 1998.

[WC95] G. D. Walborn and P. K. Chrysanthis. Supporting semantics-based transaction processing. In *Proceedings of the 11th Symposium on Reliable Distributed Systems*, pages 31–40, September 1995.

[WC97] G. D. Walborn and P. K. Chrysanthis. Pro-motion: Management of mobile transactions. In *Proceedings of 11th ACM Annual Symposium on Applied Computing*, 1997.

[WL95] M. H. Wong and W. M. Leung. A caching policy to support read-only transactions in a mobile computing environment. Technical Report CS-TR-95-07, Chinese University of Hong Kong, May 1995.

[WSCY] O. Wolfson, P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. To appear in a special issue of the Distributed and Parallel Databases Journal.

[WXCJ98] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: Issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 111–122, Capri, Italy, July 1998.