Group-based Data and Location Management in Mobile Environments

Lam Hoi Kit

A thesis submitted in partial fulfilment of the requirements for the Degree of Master of Philosophy

Department of Computing

The Hong Kong Polytechnic University

December 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____

Lam Hoi Kit

Abstract

Pervasive and ubiquitous mobile applications have sprung following the proliferation of wireless communication devices. With the ability to detect the positions of mobile devices, value-added location-dependent services can be offered to mobile users by taking advantage of a user's location information. Location management is crucial in supporting location-dependent applications. While location management is relatively straightforward in infrastructures that allow the physical tracking of mobile client location at the server, there are challenges for systems in which mobile hosts have to keep track of their own locations, reporting to the location server themselves. Location-dependent queries are then issued from location-dependent applications to the location server, resolving the geo-referencing matters for result generation and filtering. We focus in this research on systems in which mobile clients detect and report their own location with devices like GPS. Noticing that the majority of computation and data storage tasks are assigned to servers due to the resource limitation in mobile devices, we need to address the complementary data accessing need from servers residing over the fixed network, in enabling mobile application development.

We identify data accessing and location management to be complementary and yet competing for the expensive uplink channel for sending in data requests and location information to servers. It is crucial to reduce the aggregate communication overhead, owing to the high cost of uplink bandwidth for data accessing and location reporting. Traditional research efforts are dedicated to reducing the communication overhead while preserving the system performance. However, most research works are based on the client/server model. In this thesis, a group-based model is defined by taking advantage of integrating mobile ad hoc network with the client/server model. A group-based framework that provides supports for data and location management is proposed. To realize the framework, a dynamic group formation algorithm is developed for clustering together mobile hosts with similar mobility into groups. Additional group management mechanisms are designed to cater for mobile hosts joining, departing and changing groups. The group functionalities are basically supported by the group leader, for which appropriate leadership maintenance schemes are proposed. Two applications of the group-based framework were explored, namely, group-based location management and group-based data management schemes. Performance study is conducted for evaluating the effectiveness of several variants of group-based location and data management schemes. The results demonstrate the advantages brought about by the group-based paradigm, in reducing the communication cost.

Acknowledgements

I thank very much for the guidance and counsel from my two supervisors Dr. Hong Va Leong and Dr. Stephen Chi Fai Chan. They provide valuable advice, comments and suggestions in the course of my study. Their insights during our meeting always bring up new exposure and directions that I would have missed. Meanwhile, my technical knowledge has been deepened and widened in their kind supervision.

Having the opportunity to listen the past study experiences from Ken C. K. Lee , who is currently studying his PhD. in the United States and previously got a M.Phil. in the Hong Kong Polytechnic University with the supervision from Dr. Hong Va Leong, I learned a lot from him. His help and kind support are unforgettable. I also thank for my classmates, Teddy Chi Yin Chow and Jing Zhou, who are also supervised by Dr. Hong Va Leong. Through the collaboration and discussion between them, many interesting research ideas were brought up. These have a great help and result in positive effect for each others' studies. I would also like to express my thanks to all classmates and colleagues in the Department of Computing.

Finally, I would like to thank my family with their unconditional love. This thesis is dedicated to my family for their endless support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Pervasive and ubiquitous mobile applications have sprung following the proliferation of wireless communication devices, as hardware cost continues to drop. With the ability of detecting the position of mobile devices, value-added services could be developed by taking advantage of the location information available. It is anticipated that location-dependent or context-aware applications will likely contribute to the set of killer applications in this decade. Location-dependent queries (LDQs) would be issued by mobile devices. For example, a user of a mobile device may want to know about the nearest Chinese restaurant from his/her current position. These killer LDQ applications would be difficult to be developed without the adoption of effective and efficient schemes in both data and location management.

Data and location management are fundamental services to be provided by a mobile computing environment in order to support higher level applications, including location-dependent applications. Efficient data accessing methods and location tracking techniques are critical for the success in developing a mobile computing system that supports location-dependent query. A mobile environment is composed of a set of mobile hosts moving around in an area served by a base station or an infrastructure of base stations, interconnected by wired networks. Mobile networking environments differ in the communication infrastructure, which in turn leads to the development of

varying techniques by researchers in accessing data and delivering the location management service. These environments are almost always based on the client/server model and is focused on the communication between the mobile hosts and the fixed server. Efficient utilization of the asymmetric bandwidth between the hosts and the server is often the key focus. Another research focus in mobile computing is mobile ad hoc network, which is concerned about the communication between mobile hosts in a mobile computing environment, in which the network topology changes continually. Reliable and efficient message routing to the destination becomes the most important issue.

1.1 Individual-based Data and Location Management

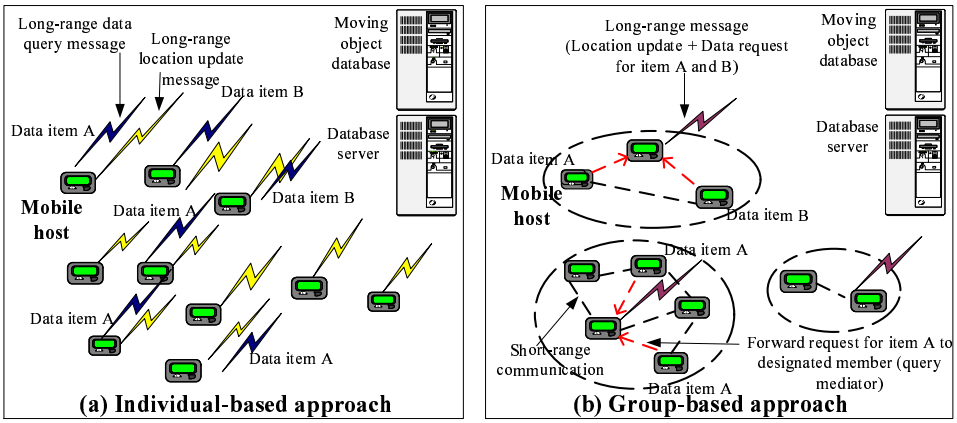


Figure 1.1: Individual-based approach and group-based approach in data and location management.

In traditional client/server based mobile systems, each mobile host heavily relies on expensive uplink channel for data and location management on an individual basis. We term such an approach of data and location management the *individual-based approach*, as depicted in Figure 1.1(a). In the location management context, each mobile host reports its own location information individually to a location server. A location server is normally established to maintain information about the current location of each mobile host. This functionality is often supported in the backend by a moving

object database [55, 56], which keeps track of and maintains the indices on the locations of the mobile hosts. Mobile hosts report their location information to the moving object database according to the location updating strategy adopted. The moving object database maintains the location information of mobile hosts for answering the queries about the interested mobile hosts. Quite often, spatial databases [16] or their variants are adopted as the basis of a moving object database. Similarly, in mobile data management, data requests are generated from each individual mobile host to stationary database servers. As depicted in Figure 1.1(a), there are 10 mobile hosts in the mobile environment and each host reports its location to the location server individually. In total, 10 location update messages are generated to the location server and 6 data request queries are sent to the stationary database server. Some of them involve requests on overlapping set of data items but each has to make the request individually. As a result, mobile hosts nearby report similar or approximately the same location information to the location server and probably issue duplicate requests of same data items to a database server, by using the individual-based approach.

It is obvious to observe that one major problem in individual-based approach is that the demand of expensive uplink channel becomes overwhelm when the mobile host population becomes large. Most existing research works are proposed to alleviate the problem of uplink bandwidth consumption by tuning down the frequency of utilizing uplink channel and studying the impact to the query precision [8, 32, 56] for each mobile host. For a mobile environment with mobile host population M and the frequencies of location updating and data access by a mobile host being f and $\check{q}$ respectively, the expected demand on the precious uplink channel in terms of number of uplink messages in a unit time is $M(f + \check{q})$. Both host population factor, and location updating and data access frequency factor are of equal importance in contributing to the demand on the uplink channel.

1.2 Group-based Data and Location Management

Few research works are proposed on developing schemes that can scale with respect to the mobile host population in location update and data access. With the emergence of short-range communication technologies like Bluetooth, mobile ad hoc networks [18] can be established and mobile hosts can collaborate, in much a similar manner as a peer-to-peer (P2P) network. In our viewpoint, it is possible and yet practical to group nearby hosts so that hosts can interact within a short transmission range, and let one of the hosts in the group report the location of the whole group representing all hosts within. Query requests of the same or similar data items can be aggregated together and issued by one of members in the group. This query aggregating and requesting jobs can be distributed among the members in the group. Two interesting example applications can be illustrated. First, in the Advanced Traveler Information System (ATIS) [12], to monitor the locations of vehicles of interest, vehicles in a highway segment can be clustered so that one representative vehicle, rather than all, can report the locations on behalf of the group within the segment. We call each such cluster of moving objects a *group*. Furthermore, in tourist supporting systems, tourists with common point of interests often move together as clusters and query similar information about the scene visiting. Therefore, different groups can be established for tourist supporting purpose. Location of different tourists within a group can be aggregated into single location report to the location server so that the corresponding tour guides are able to ensure the safety of the tourists. Additionally, similar data requests can be batched within a group. The consolidated query is sent to the database server by one of the members in the group. As illustrated in Figure 1.1(b), we can observe that there are 3 groups formed, so that only 3 location update messages are issued to the location server, instead of 10 messages from individual hosts. The number of query messages is also decreased from 6 to 3 when the group-based model is applied in data query.

To capitalize on this observation in costly uplink message reduction, we propose a *group-based approach* for data and location management, as shown in Figure 1.1(b).

In general, nearby hosts with similar mobility [22] are grouped or clustered together to form *groups*. Within a group, one host is selected as a *leader*. It is responsible for managing the membership of mobile hosts within the group and updating group location to the location server.

Upon the model, a Group-Based Location management scheme is proposed, in short "GBL". The leader of the group is dedicated to reporting the location of the group. Meanwhile, other hosts in the group report their individual locations to the leader. The location information is aggregated within a group and the leader reports to the location server. Let g be the number of groups formed in the environment with a mobile host population of M. The number of uplink messages generated in a unit time then becomes gf, instead of Mf in an individual-based approach.

Another application of the model is Group-Based Data management (GBD), overlapping queries could be consolidated and eliminated by sending queries from group members to their group leaders instead of sending directly to server. Leaders collect the queries from members over a time period and perform query consolidation for those received queries before forwarding the consolidated queries to the server. Supposing that the average number of queries from members to be consolidated within a group in a unit time is d, the number of uplink query messages in a unit time in the group-based model becomes $\frac{M\check{q}}{gd}$ while that in the individual-based is $M\check{q}$.

Reducing server load and long-range communication traffic (by limiting the number of uplink update and query messages to stationary servers) are not the only benefits brought about by the group-based model; members in a group are also not mandated to reserve bandwidth for location updating to the server constantly with the group-based model. A direct and positive consequence is that mobile hosts no longer need to possess the communication capability with the remote server. Group-based model allows only a portion of mobile hosts in a group being equipped with long-range communication functionality. Short-range communication-only mobile hosts can issue query requests to database server through other long-range communication enabled mem-

bers in the group. Hence, scalability is increased. Since the location of a mobile host is different from time to time, maintaining very high precision of location information stored in the location server is almost impossible, even in the case of conventional individual-based location updating [56]. It is acceptable to lower the information accuracy with saving communication cost in GBL. Another negative consequence of group-based data and location management is more local communication overhead being introduced. More local traffic is generated for the group management and leadership maintenance purposes. With effective group management and leadership maintenance strategy, the increase of mobile host load and intra-group communication traffic in group-based data and location management can be leveled out when the power consumption involved for intra-group communication and computation is small comparing with long-range communication.

The main contributions of this thesis are as follows: (a) We propose a group-based paradigm to tackle the problems related to location and data management in mobile environment. The problems are handled via the design of data and location management schemes which aim at saving precious uplink traffic at the expense of increased need on cheaper local communication. (b) A group-based framework is designed for development of mobile applications in group-based paradigm. It is expected that the benefits of the group-based framework become more significant when more mobile applications are built on top of it. (c) To realize the group-based framework, we propose a set of mechanisms for group formation, membership management and leadership maintenance with the consideration of local communication cost and the instability of a group, due to movement of hosts. (d) We study the performance of our proposed group-based location and data management schemes via simulation. They are shown to improve the performance in terms of the number of uplink messages and aggregated communication cost.

1.3 Outline

This thesis is organized as follows. Chapter 2 gives a survey on the related research in location management, mobile data management and mobile ad hoc network, as well as the relevant issue on leader election in mobile environment. The group-based model is defined in Chapter 3. In the chapter, a group-based framework is also designed. Discussion on the group management in group-based mobile environment can be found in Chapter 4, which includes dynamic group formation scheme and basic membership maintenance strategy. Based on the our preliminary study on the basic membership maintenance, improvement strategies in group joining are proposed and discussed. In Chapter 5, leadership maintenance in the group-based environment is investigated and a leadership maintenance scheme is presented. To demonstrate the application of the group-based model and the group-based framework, a group-based location management scheme is developed in Chapter 6. Simulation study on different mechanisms to support the framework in the GBL context is also conducted for evaluating the effectiveness of the group-based location management scheme. Another application of the group-based framework, which is known as group-based data management scheme, is studied in detail in Chapter 7. The effectiveness of the scheme is investigated via simulation study. Finally, we conclude our work and present possible future directions of the research in Chapter 8.

Chapter 2

Related Work

Data and location management research in mobile computing environment [27] has been an active research topic in mobile computing area. Traditionally, data management research is focused on the mobile environment between mobile host and fixed network. It is because mobile clients are almost always scarce in storage capacity, network bandwidth, processing and battery power. Complex computation job and storage can only be empowered to a fixed network server. Based on this specific characteristic of mobile client/server communication environment, various techniques [1, 8, 9] have been developed for enhancing the system performance in both location and data management perspective.

Another aspect of research in mobile computing area is mobile ad hoc network. In mobile ad hoc network, a fixed infrastructure is not required. Network topology and configuration always change. The research works are focused on how to provide network services by considering the cooperation between mobile hosts. Although there is a substantial difference in the mobile environments that these two research directions are focusing on, it is believed that research works on combining these two different environments can be beneficial in developing mobile applications [23, 24, 42]

In this chapter, different technologies, schemes and techniques in mobile data and location management research areas will be discussed. Topics include wireless

communication technologies, leadership maintenance, location management and, data management.

2.1 Wireless Communication Technologies

Wireless communication technologies contribute in the critical component in enabling the development of mobile and wireless computing. These communication technologies can be divided into two different categories. With the limitation in processing power, mobile computer systems almost always follow the client/server paradigm. With client/server paradigm, large volumes of data could be stored on remote servers and complex computation process could be accomplished by application servers. The ability to communicate with a remote server is important in this paradigm. The network model developed based on this paradigm can be categorized as *infrastructure-based network model*. On the other hand, emerging short-range communication technology enables communication between devices. This leads to another area of research in wireless communication and mobile computing. The network model in this category is known as *mobile ad hoc network model*. In this section, technologies involved in both network models will be discussed.

2.1.1 Infrastructure-based Network Model

In an infrastructure-based network model, a base station or a set of base stations provide the wireless communication service to the mobile hosts which are moving around. Each base station provides service within a bounded area, termed as service area. As shown in Figure 2.1, the dotted ellipse surrounding each base station represents the service area that the base station can serve. Typically, infrastructure-based network model is based on client/server communication model. A mobile host can be served within the service area of a base station. Those base stations are always assumed to be connected with the fixed network. Therefore, client/server model is popular in

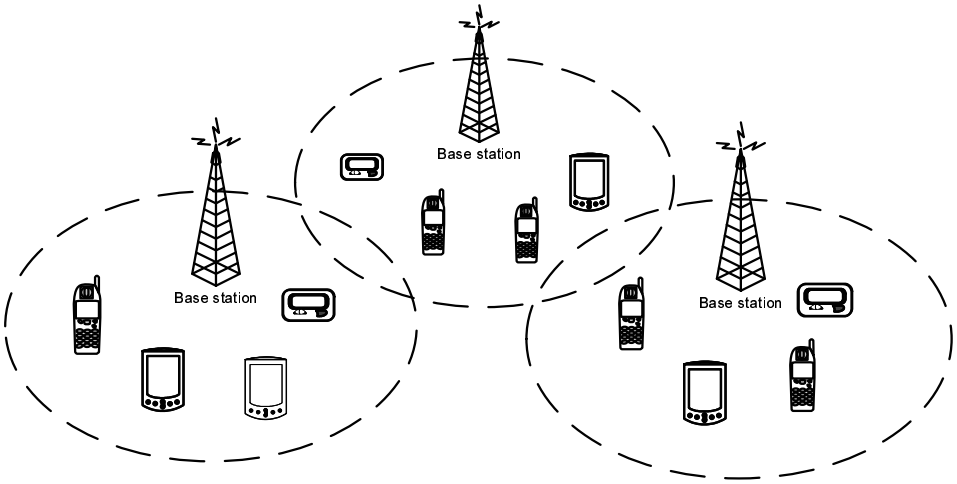


Figure 2.1: Infrastructure-based network model.

developing mobile applications. Application servers, which reside on the fixed network, provide the corresponding services to the mobile clients. A main reason that client/server communication model is attractive in the mobile computing environment is that mobile clients are always limited in processing and storage capabilities. In such a mobile environment, asymmetric bandwidth is assumed. It means that there is a high cost in using uplink channel and only a low cost in using downlink channel.

Existing wireless communication technologies often follow the infrastructure-based network model. In personal communication network (PCS), cellular architecture is employed. Clients make calls with another through the base station. Similarly, in the wireless LAN (WLAN), mobile hosts connect with other machines or mobile hosts for communication through the access point, which resides on the fixed network and is connected with the Internet.

2.1.2 Mobile Ad Hoc Network (MANET) Model

Ad hoc mobile networks are developed for scenarios where it is not always convenient and flexible to construct a *virtual* centralized communication network infrastructure, in which centralized infrastructure and configuration is not required. The communication network is established in an ad hoc manner and is self-organized among the mobile hosts participating in the network. Popular technologies in supporting mobile ad hoc

network model are Bluetooth [17] and IEEE 802.11. Research issues in networking aspects include routing in an ad hoc network [30, 48], power management [58] and location management [40, 57].

In particular, research work on providing a relatively stable layer of network on top of flat ad hoc network routing forms a major research focus, and in the context of group-based paradigm, mobile host clustering in the ad hoc network into sets of groups. Popular schemes proposed include lowest-id and highest degree heuristics [11]. Clustering based on mobility prediction was also studied [44]. In [44], an (α, t) framework was provided for forming clusters with probability α of link availability between all mobile hosts in a cluster in time interval t. In addition, an on-demand distributed weighted-based clustering algorithm [11] is proposed, by considering several factors that will affect the stability of a cluster. A mobility-based clustering algorithm [4] is developed in which the mobility of a mobile host is considered. However, the research focus is on providing efficient routing service in a relatively stable clustered network environment. The mobility similarity of mobile hosts is not the main concern for providing the service. As a result, the existence of clusters overlapping in communication range is not a desirable property in those algorithms. However, allowing overlapping clusters in terms of communication range is a desirable result in the context of location reporting service if there are several sets of hosts possessing different mobility patterns in a particular physical area. The group mobility movement model, called Reference Point Group Mobility (RPGM) model is proposed in [22]. It considers only the physical location factor for providing another category of movement model for evaluating the effectiveness of different ad hoc network protocols. Another movement model called Reference Velocity Group Mobility model is proposed in [53]. It is an extension of the RPGM model with consideration of the velocity factor. A distributed sequential clustering algorithm, which is adopted from pattern recognition, was also proposed. Dynamic group discovery strategy for forming groups for routing in ad hoc network was proposed in [21]. However, a relatively ideal group-based movement environment is assumed in both pieces of work.

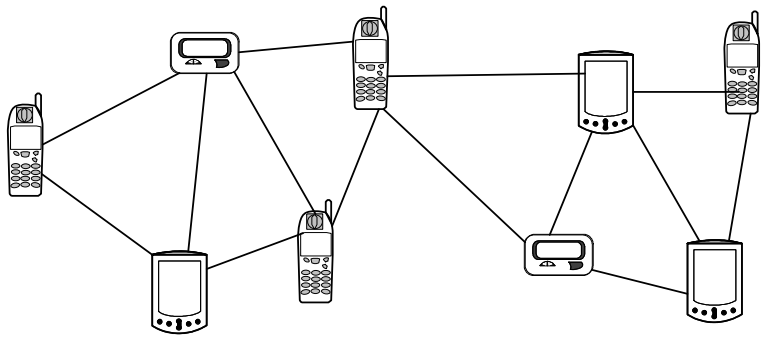


Figure 2.2: Mobile ad hoc network model.

2.1.3 MANET and Infrastructure-based Network Integration

With the integration of mobile ad hoc network into traditional client/server communication environment based on an infrastructure, host cooperation becomes possible. The strengths of mobile ad hoc communication paradigm and client/server paradigm complement each other. Such a model can be treated as a hybrid model [50]. As mentioned in [50], a mobile device can still enjoy the services outside the service area of the infrastructure-based network through the ad-hoc mobile network. Furthermore, more types of mobile devices can be supported in this model, hence system compatibility increases.

Several research works [23, 24, 42] have been conducted for investigating the issues in integrating cellular infrastructure-based network model with mobile ad hoc network model. The throughput and energy consumption is evaluated. In [42], it proposed a Unified Cellular and Ad-hoc Network (UCAN) architecture, which can enhance cell throughput. A mobile client is equipped with two wireless interfaces, one for cellular and another one for ad hoc network. Mobile clients would act as a proxy client (for communication with the base station), or a relay client (for forwarding or routing data packets to destination client). Based on the architecture, protocols for proxy discovery and routing are proposed. The 3G base station scheduling algorithm is refined for maintaining fairness in the UCAN. In addition, a secure crediting mechanism is provided for motivating mobile clients to become favored in relaying data packets for others.

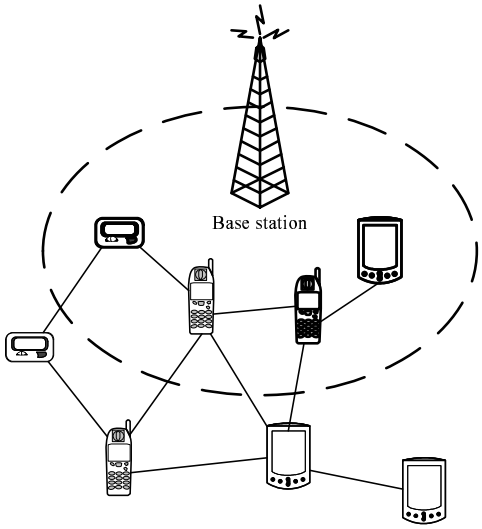


Figure 2.3: Integration MANET with infrastructure-based.

2.2 Leadership Maintenance

In mobile ad hoc network clustering, a leader may be elected among mobile hosts in each cluster, called *clusterhead*. For those clustering algorithms, leader election algorithm is always an integral part in the formation of clusters. On the other hand, there is little discussion on the issue of handing over the leadership to another host in a cluster when a leader departs its cluster in a voluntary or involuntary manner. An intuitive approach to deal with the problem is to repeat the clustering algorithm among the mobile hosts in the problematic cluster periodically or adaptively. In [25], a data replication scheme, DRAM, was proposed. The scheme is executed periodically according to the relocation period. DRAM consists of two major phases: the allocation unit construction phase and the replica allocation phase. Cluster maintenance tasks are collected in the allocation unit construction phase, including splitting groups, assigning leaders to newly split groups and forming group for the mobile hosts in a state not belonging to any group. Those maintenance tasks simply involve periodic execution of the clustering algorithm. In [11], the nodes will monitor the signal strength from the clusterhead. If a node finds that the signal strength falls below a threshold, it notifies the clusterhead while trying to move over to another cluster. If the node is not covered by any cluster, the clusterhead election algorithm is invoked.

Similar to clustering in mobile ad hoc network, virtual backbone formation [41] provides routing service in an ad hoc environment. A number of mobile hosts, termed virtual backbone nodes, participate in virtual backbone formation and maintenance. As with the role of a clusterhead in ad hoc routing, the virtual backbone nodes provide the routing service. As topology changes, structural and connectivity maintenance is required. Like leadership maintenance, mechanisms for generation or merging of virtual backbone nodes are required for handling mobility and failure of mobile hosts.

In both virtual backbone and clustering in mobile ad hoc network, failure in virtual backbone nodes and in clusterheads cause performance degradation in application or network services. The reason is that there is no leader for providing the necessary services during the period when the leader is absent. With a secondary leader in a group, the backup leader can stand by to take over the tasks from the primary leader, thus increasing system fault-tolerance.

Multi-hop leader election algorithms in mobile ad hoc network environment were proposed in [43], based on temporally ordered routing algorithm [47], which is in turn based on [15]. Two leader election algorithms were proposed. One is capable of handling a single topology change at a time, while the other handling multiple concurrent topology changes. Each node is assigned a unique “height”. Links are directed from higher height to lower height. These directed links form a directed acyclic graph (DAG) so that the destination is the single sink. Each connected partition in the ad hoc networks is a DAG. The leader in a connected partition thus is the single sink in this partition. When a splitting or a merging of partitions is detected, the leader election algorithm is invoked. As a result, a single leader will eventually be elected in a connected partition. However, performance study of these leader election algorithms is lacking.

2.3 Location Management

Extensive research works on location management in mobile wireless environment have been conducted. They are concerned with efficient ways in keeping track of the location of mobile hosts and furnishing such information upon request. Many location positioning technologies were developed [20]. One popular location positioning technology in outdoor environment is Global Positioning System (GPS). It is a satellite-based location positioning technology. Triangulation is used for locating the position of a mobile host. However, it is challenging when applying GPS in urban area, where there are many tall buildings. The accuracy and the location detecting ability are affected seriously. Other technologies using existing personal communication network, such as GSM, to position a mobile host are developed. On the other hand, various indoor positioning systems are developed to address the difficulty in using GPS in indoor environment. Technologies developed for indoor positioning include Cricket [49], the Active Badge [54], Bat [19], and RADAR [5].

Another important issue is location updating strategy, through which mobile hosts report to location server about their current location. There are two major models with respect to location management in a centralized communication model. One is adopted in the personal communication network (PCS) [8, 39, 46] and the other is based on a moving object database residing in the fixed network. The PCS model is based on an infrastructure of cellular architecture, in which mobile hosts report eagerly to respective cells, whenever a cell boundary is crossed, or reporting lazily with respect to boundary crossing. Location management is concerned with location updating strategy to balance between the location update cost and the paging cost with varying location update condition. The paging cost is the number of cells to be examined for making a call to a target mobile host from another mobile host. The precision is of the cellular granularity. Techniques and schemes for location management based on cellular architecture in personal communication network, including time-based, distance-based and selective location update techniques, were proposed. Location updating based on the

mobility pattern of a mobile host was proposed in [8]. A table with the history of the cell against the time period in which a mobile host resided is stored. Location update only happens when a new record is inserted, which was previously inexistent in the table. In order to page a mobile host, the table is searched.

In the moving object database model, it is often assumed that mobile hosts are equipped with location positioning system, e.g., GPS. Location update strategies with location prediction function for moving objects tracking in moving object database was proposed [56]. It is used for determining when a location update for mobile hosts should be triggered. The distance between the predicted location of mobile host m and the actual location of m after a certain time period t was defined as *deviation*. If the deviation is larger than a threshold D, location update will be triggered. Owing to the need to answer queries based on the location, spatial databases [16] are often required for effective location-dependent query processing. In [3, 37], the moving objects are bounded by a region where they will likely be residing in. This is called a *safe region*. As long as a moving object stays within the region, updates to the spatial database can be eliminated. In [37], the arbitrary-shaped region can be adaptively adjusted according to the movement of the objects and their movement pattern. Three different approaches were proposed for adaptive adjustment of a safe region. However, most existing research works in location management are focused on location update schemes for individual mobile host. A group-based location update scheme for personal communication network (PCS) was proposed in [26]. The scheme is based on HLR/VLR architecture with centralized database. The aim is to reduce the number of location registrations by grouping a set of mobile hosts at their serving VLRs and to reduce the cost for signaling in location management with slightly increase in paging cost. However, it is difficult and often impractical by assuming that all hosts or objects move in the same direction as in [26], owing to the highly dynamic movement nature of mobile objects. Hence, the stability of a group will be degraded by frequent join group and leave group events triggered by mobile hosts.

Regardless of the communication structure, the majority of traditional location up-

dating strategies are based on the client/server communication model. Large volumes of expensive uplink traffic will be generated for location reporting by a large host population to the fixed server. In this thesis, we proposed a group-based location management scheme (GBL), which will be discussed in detail in Chapter 6. The scheme takes into advantage the integration of mobile ad hoc network with traditional client/server communication structure to reduce the uplink traffic and workload of servers.

2.4 Data Management in Mobile Environment

Efficient and effective ways for accessing data in a highly constrained mobile environment is essential for the success of mobile computing systems and applications. Conventionally, the research works on mobile data management focus on providing effective solution in the mobile client/server environment. Because of asymmetric communication characteristics in the infrastructure-based network model, extensive use of uplink channel should be avoided. It is necessary to reduce uplink communication through various approaches, by utilizing the downlink channel and available local resources optimally. This leads to extensive research works in mobile/wireless data broadcasting and data caching.

2.4.1 Data Broadcasting and Caching

In mobile communication environment, asymmetric environment is always assumed. It means that mobile hosts have to communicate with remote server in fixed network through low cost downlink channel and expensive uplink channel. To take advantage of the broadcast nature of low cost downlink channel, dissemination of data from remote servers to mobile hosts is often accomplished through data broadcasting [1, 28]. The advantage of data broadcasting is to provide data delivery service to a large scale of mobile users through a less expensive downlink channel. The basic heuristic or rule of thumb in mobile data broadcasting is to deliver those data items that most of mobile

users are interested in. Those popular data items are termed as hot data items. For those not-so-popular data items, a better method of data access could be achieved perhaps by sending a direct uplink request to the server.

In data broadcasting, a broadcast program [1, 35, 36, 38] is constructed and data is delivered to mobile hosts sequentially according to the broadcast program generated. Different data broadcasting schemes were developed to provide effective ways to disseminate data from servers to clients. Two performance metrics are adopted in measuring the effectiveness of a data broadcasting scheme, i.e., *tuning time* and *data access time*. Tuning time means the amount of time that a client is active in listening the data broadcast channel for requested data items, while data access time means the time period required to gather all the requested data items. To reduce the access time, more popular data items could be broadcast more frequently. In [1], data items will be broadcast with different frequency based on the popularity of data items in the mobile environment. The more popular data items there are, the higher broadcast frequency for broadcasting the data items. As a result, mobile hosts are able to get the data that they want earlier. Because of the sequential data delivery nature in data broadcast, in the worst case, mobile hosts have to wait for the whole data broadcast cycle for to get the data items they want.

With data broadcast, mobile hosts have to stay in active mode so that unnecessary resources are consumed, especially in energy saving perspective. This brings a negative impact to the tuning time performance metric. To tackle the problem, indexing technique [28, 38] is developed for reducing a mobile host's active time in listening for the broadcast program. Data index is generated and becomes a part of data broadcast program, normally at the beginning of broadcast data segments. Those data indices can be treated as the summary of data being broadcast in the broadcast cycle or segment and the estimated time of the data to be delivered. Mobile hosts examine the index to check for the availability of their interested data items and their broadcast time. After that, mobile host can turn to the doze mode until the broadcast time. In [38], the data items are scheduled as a broadcast program. In each data segment, signatures for

those data items are generated and these data signatures are superimposed to form an index for that data segment. Then all indices will be put in the front of the broadcast program or replicated within it. The server will then broadcast the whole broadcast program to the mobile hosts. When a mobile host issues a query, the condition criteria in the query will be transformed as a signature, known as query signature. The mobile host will actively listen for the index to check for the wanted data item arrival time by comparing the query signature with the index. The mobile host will then turn to a doze mode and wake up until the wanted data items appear on the broadcast. Semantic data broadcasting scheme was proposed in [35, 36]. In the scheme, data stored in a database is partitioned into a set of chunks, which are treated as semantic partitions, according to the semantics of client queries initiated. Similar to [38], each broadcast program is divided into two parts, namely, semantic index and a list of chunks.

To further reduce the necessity of interacting with remote server while maintaining reasonable performance in data accessing, data caching [6, 9] is another approach for eliminating the number of expensive uplink queries from mobile host to server. The general concept of data caching is that each mobile host allocates a portion of memory resource, called cache, for storing the results of query being recently asked. When a mobile host generates a new data query, the cache is examined to check whether there exists the result of the new query or not. If the query can be answered by the cache content, it is said that there is a cache hit and no further uplink query message will be sent to the server. Data access time is also reduced because the query can be answered from the cache. Otherwise, there is a cache miss when the current contents cannot be used to answer the query. If that is the case, the query will be sent to the remote server for proper response. The newly replied data items will then be stored to the local client cache. Since the size of a cache is limited, it is possible that the cache becomes full. Cache replacement policy is developed for replacing less suitable cached data by the new query result, in order to maintain a reasonable hit ratio. On the other hand, the cached data may be out-of-date, so invalidation technique is also required.

Various data caching schemes are proposed in mobile environment [34, 38]. Al-

though traditional database-oriented data caching schemes could be employed, those schemes are server-filtering schemes. Such schemes may not be good for mobile environment because they often lack the semantics about the cache content. A mobile client does not have any hints or information about the cache content whether the query could be answered entirely from the local cache or not. The client needs to send the query request to the server database for possible additional data items. These further contacts to the server are not desirable and data caching schemes for mobile environment are proposed for addressing the issue.

The basic idea of semantic query caching [34] is to answer client's query from cache storage as much as possible according to the semantics of the query. The remaining unanswered portion of data is requested from the server. According to the semantics about the cache content, it is easy to know whether the query can be answered entirely from the cache or only partial query can be answered from the cache. Basically, when a mobile host issues a query, the semantic cache content is examined. Two queries will be generated. The first one, called probe query, is the query that is answerable from the cache. The second one is supplementary query; it describes the data items missing, to be answered from the server. If a query can be entirely answered from local cache storage, unlike server-filtering schemes, no further uplink request is required for checking which data items are not available in the local cache storage in semantic caching. The demand of expensive uplink traffic is reduced.

Another data caching technique is called signature-based data caching [10, 14, 38]. In signature-based data caching, a bit vector is generated for each data item. For a block of data items, the signatures of data items are superimposed into a single signature for that block. When a user wants to access one or more of data items, the signature of query is also generated. The query signature will then be sent to the information server for retrieving the answer of the query. Signatures of the blocks are compared with the query signature. The two signatures match if all bit positions that are set to "1" in the query signature and the corresponding bit positions in the block signature are also "1". If both block signature and query signature match, the block

will potentially contain the data items the query wanted. With signature-based data access, false negative cannot happen but it is possible to have false positive. With false positive, a block may be considered containing potential answer for the query but it actually contains no data item that the query requests. With good choice of hash function for generating data signature, the probability of false positive could be low. Signature caching also has an advantage in which the techniques could be generally applied to various kinds of information media.

2.4.2 Mobile Query Processing

New query processing issues are introduced in mobile environment, especially in location-dependent query processing [27, 51]. New query operators would be required for supporting the criteria of querying spatial related data [51]. Mobile computing systems have to handle the situation that a mobile host issue a query when it is still moving. It is possible that the mobile host migrates from the original service area to another service area in the infrastructure-based network model. In [52], five strategies are proposed for processing three types of queries issued by mobile hosts, namely, location queries (queries about the location of a mobile host), existence queries (queries determining a mobile host is currently active or not) and data query (queries for information from a mobile host). Based on the system model formulated in [52], the issue of mobile host migration could be addressed by the collaboration between query servers, which handle mobile queries in the global network, and mobile host servers, which are responsible for the communication between global network and mobile hosts through their wireless interfaces. Since mobile host could also be a data source, approaches to process queries with data spreading among different sites, including stationary server and mobile hosts, would be another issue. Different approaches from traditional distributed query processing [31, 33] may be required in handling the unique features in mobile environment. In database context, mobile hosts also store data in relations. Procedures and algorithms for processing query with different data source, involving

stationary server and mobile hosts, are studied in [33]. In the paper, two join schemes and query processing schemes for multi-join queries are proposed. The focus is on the mobile distributed query processing with join operations. On the other hand, semantic query cache [34] can also be treated as a query pre-processing scheme for a mobile host to execute before sending to a query server to eliminate sending unnecessary queries which can be entirely answered by local semantic cache. However, those works are focused on processing queries issued from a single mobile host individually. Queries from other mobile hosts in the vicinity are not considered. From another point of view, similar queries would be generated from the mobile hosts nearby. Further pre-processing work could be performed for reducing multiple similar queries. In this thesis, we proposed a group-based data management scheme (see Chapter 7) that mobile queries could be consolidated within the group before issuing them to the server for reducing the demand of uplink traffic in a mobile environment.

Chapter 3

System Model and Group-based Framework

This chapter provides the formulation of the system model used. A dynamic group formation scheme, which is invoked in the system activation, is developed according to the model (see Chapter 4). Based on our system model, a group-based framework is proposed for mobile computing system development in group-based approach.

3.1 System Model

In the group-based system model, each mobile host m is assumed to possess a unique ID and a location positioning system (e.g. a GPS sensor) for monitoring its existing location and keeping track of its movement information. The current location of m is denoted by $\langle x_m, y_m \rangle$, while the movement information is maintained and represented as a vector $\vec{v}_m = \langle v_{xm}, v_{ym} \rangle$, being resolved into the x and y components, as shown in Figure 3.1(a). In addition, the latest updated location for m and the latest updated velocity are denoted by $\langle x_{um}, y_{um} \rangle$ and $\vec{u}_m = \langle u_{xm}, u_{ym} \rangle$ respectively. Thus, for a predefined time parameter τ as shown in Figure 3.2, m's predicted location after time period τ is $\langle x_m + v_{xm} \times \tau, y_m + v_{ym} \times \tau \rangle$ and if information from the latest update

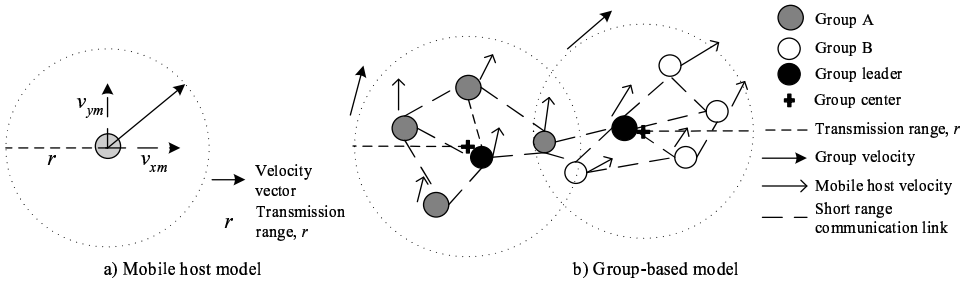


Figure 3.1: The system model.

is available, m's predicted location for a time period from the latest updated time t_{um} to now ($t_{um} < now$) can be estimated as $\langle x_{um} + u_{xm} \times (now - t_{um}), y_{um} + u_{ym} \times (now - t_{um})\rangle$ [2]. Two mobile hosts are considered as *neighbors* if the Euclidean distance between the two hosts is smaller than the transmission range of these two hosts (i.e., they can communicate in an ad hoc mode).

In addition to a conventional long-range wireless network (i.e., the downlink channel for information delivery and the uplink channel for location reporting and query submission), a mobile ad hoc network is assumed in our model. In the mobile ad hoc network that connects all (almost all) mobile hosts, hosts maintain wireless links with one another within a constant transmission range of ***r***. Very often, the transmission range is expressed as an Euclidean distance. Groups are formed by clustering together sets of nearby mobile hosts. In other words, the ad hoc network is conceptually split into potentially overlapping clusters. Each cluster is called a *group*, each of which has a *leader* associated. The leader of a group is responsible for managing group activities like *member join* and *member leave*.

3.1.1 Properties of a Group

In the group-based model, a group is a natural collection of mobile hosts that can communicate with one another and that move together in an aggregated manner; a leader can be elected from a group to act on behalf of the group. Thus, to qualify as a potential member to a group, G, a mobile host m should be at most a distance of ***r*** away from the group location. The location of a group, G, refers to the center of

the circle that is $\langle x_G, y_G \rangle$, where $x_G = \frac{1}{|G|} \sum_{m \in G} x_m$ and $y_G = \frac{1}{|G|} \sum_{m \in G} y_m$, and the movement of G is represented as $\vec{v}_G = \langle v_{xG}, v_{yG} \rangle$, where $v_{xG} = \frac{1}{|G|} \sum_{m \in G} v_{xm}$ and $v_{yG} = \frac{1}{|G|} \sum_{m \in G} v_{ym}$. Similar to the notations adopted for mobile hosts, the latest updated location and velocity of the group G are denoted as $\langle x_{uG}, y_{uG} \rangle$ and $\vec{u}_G = \langle u_{xG}, u_{yG} \rangle$ respectively. Thus, the predicted location of G after time τ, as shown in Figure 3.2, is $\langle x_G + v_{xG} \times \tau, y_G + v_{yG} \times \tau \rangle$ and predicted location of G from the latest updated location with latest updated time t_{uG} at now is $\langle x_{uG} + u_{xG} \times (now - t_{uG}), y_{uG} + u_{yG} \times (now - t_{uG}) \rangle$. The network topology is illustrated in Figure 3.1(b), in which there are two groups **A** and **B** formed. The movement of the two groups, the group leaders and the individual group members are also shown. We call a group with only one member a *singleton group*, i.e., the sole member is the leader itself. The host in a singleton group will perform the group finding process periodically based on a predefined location sampling period, τ_s, until another group is found for joining or another host considers joining this singleton group.

3.1.2 Degree of Affinity

In order to maintain a more stable group, members within a group should be similar in term of mobility. We define a notion of *degree of affinity* to measure the movement similarity between mobile hosts or groups, which we term *mobile domains*. Hereafter, a mobile domain denotes either a mobile host or a group. We adopt the degree of affin-

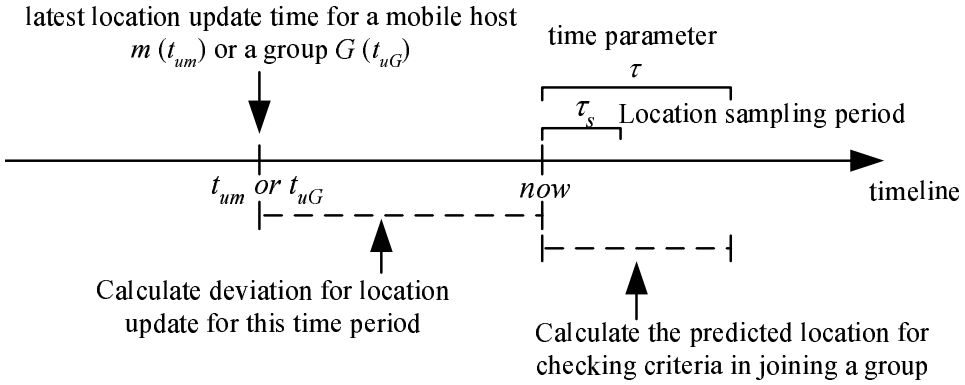


Figure 3.2: Timeline to illustrate the different notations on time.

ity as one most important factor to be considered in leader election (see Section 4.1). The higher the degree of affinity of a mobile host to its neighbor, the higher potential is the mobile host to become a leader of the group. The other factor to be considered is the degree of connectivity of a potential leader. The higher the connectivity, the more beneficial is for the host to become a leader of the group. Thus, highly stable groups could be formed with high movement similarity and connectivity within the groups.

The degree of affinity is also useful in helping to find the most suitable group to join for a mobile host and calculating the next local update deviation threshold. The degree of affinity, $s_{j,k}$, between two mobile domains, j and k, is defined by the equation:

$$s_{j,k} = \alpha(1 - \frac{dist(j,k)}{r}) + \beta(1 - \frac{\sqrt{(v_{xj} - v_{xk})^2 + (v_{yj} - v_{yk})^2}}{\sqrt{v_{xj}^2 + v_{yj}^2} + \sqrt{v_{xk}^2 + v_{yk}^2}}),$$

where $\alpha+\beta = 1$. The distance $dist(j,k)$ is the Euclidean distance between two mobile domains j and k. The distance contribution measures the normalized distance between the mobile domains. The speed contribution measures the "normalized" difference of the two movement vectors of the mobile domains against their total length.

There are two types of factors in the definition of degree of affinity: distance factor and movement factor. The distance factor is reflected by the distance between the locations of two mobile domains. This factor is defined as $1 - \frac{dist(j,k)}{r}$. The value will decrease as the distance between two domains increases. The movement factor is defined by the second part of $s_{j,k}$, $1 - \frac{\sqrt{(v_{xj}-v_{xk})^2+(v_{yj}-v_{yk})^2}}{\sqrt{v_{xj}^2+v_{yj}^2}+\sqrt{v_{xk}^2+v_{yk}^2}}$. First, it determines the magnitude of the vector difference between the velocities of two mobile domains. Second, it determines the similarity in direction of movement of two mobile domains. The higher value in the difference between two velocities, the lower degree of affinity is obtained. This is reflected by the speed contribution and the value is bounded between zero and one.

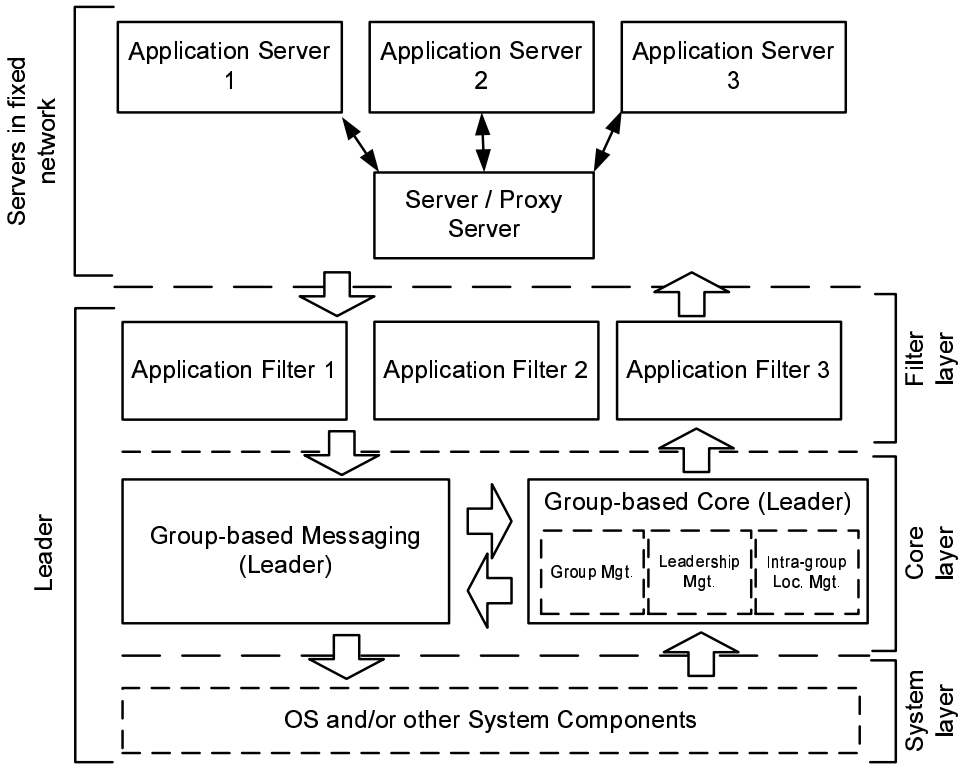


Figure 3.3: Group-based framework – Modules for leaders.

3.2 Group-based Framework

This section presents the group-based framework for facilitating the development of mobile applications with group-based paradigm. The framework abstracts the group management in mobile environment. Mobile application developers do not need to bother with those group management issues. In other words, the functionalities for group formation, membership and leadership maintenance are encapsulated in the framework. A goal of the framework is to provide a simple, easy-to-plug and standard-following interface for application developers to develop mobile application in group-based paradigm. Additional efforts for realization of the group-based model and collaboration with the framework are minimized so that developers could focus on the application logic in the mobile application development.

The group-based framework consists of several modules for supporting the group-based model we defined. As depicted in Figures 3.3 and 3.4, the core modules include

the components for group formation, membership and leader maintenance. The group messaging module is responsible for message passing between mobile hosts, and between mobile hosts and fixed network servers. Since a mobile host can be a member or a leader, some modules in the framework can be further divided into components for leaders and members. However, each mobile host should include the whole framework in developing mobile applications, except when the host does not possess long-range communication capability and cannot qualify as a leader. The detail about each module will be described later in this section.

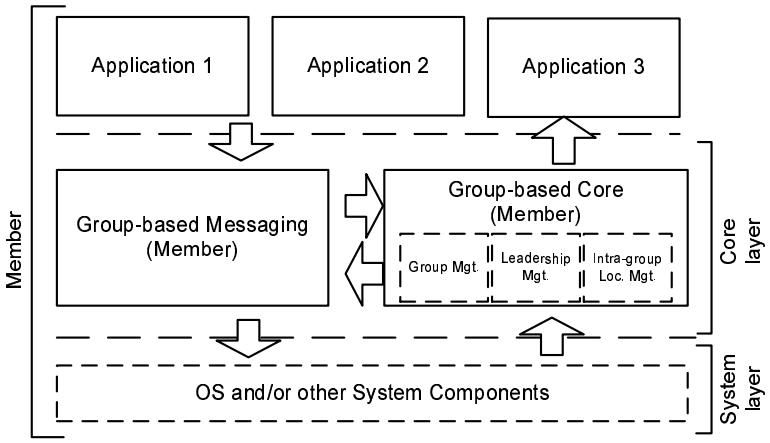


Figure 3.4: Group-based framework – Modules for members.

The group-based framework is built on top of the system layer, which includes the OS and other basic system modules. By following our group-based model, location awareness capability is assumed in the underlying system layer. Intra-group location management component is also a component in the core module. It contributes to the local level of the group-based location management scheme. The local level of the group-based location management scheme is concerned about the location updating strategy between members and leaders of their groups. The location management component forms the basis for the group location management scheme, which involves both local and global levels of location management. Details about both local and global levels of the group-based location management scheme will be discussed in Chapter 6. The component is responsible for local location update and intra-group location update. Thus, the group-based location updating scheme for reporting the group

location to a location server is a direct application of the group-based framework for studying the effectiveness of the framework.

In the core module, the job focus is on the group and location management. Both issues are the core in realizing the group-based model in a mobile environment. Group management consists of group formation, membership and leadership maintenance. Upon system activation, mobile hosts move freely and independently. No interaction exists between mobile hosts. To form groups in the environment, the group formation component provides the mechanism to cluster a set of mobile hosts with similar mobility into groups. After groups have been formed, mobile host in a group may leave and join another group. Membership maintenance component is responsible for the join and leave events from the members. Similarly, a leader may also choose to depart from an existing group. Leader maintenance component provides mechanisms for smooth handover of the job from the old leader to the new leader. Since location awareness is assumed in our group-based model for forming groups with high similarity in mobility, the local level of location management is also included in the core module.

The main job of group messaging module is to consolidate the messages from mobile applications and the core module into a single message before message passing to corresponding entities. The messaging module is responsible for allocating messages to appropriate communication channels (e.g., direct long-range message to server or short-range message to the group leader). The aim is to reduce the number of messages, hence reducing the local communication overhead. Commands or request messages will be sent to the leader by using the application programming interface (API) provided in the framework. After the message is passed through the group messaging module, the information that the core module maintains will be piggybacked. When the group leader receives a message from a member, the leaders' group messaging module will try to identify and extract different segment of the message and pass the corresponding segments to appropriate application filters or components in the framework. In the design of group messaging module, Pipes and Filters design pattern [7]

could be applied.

An application filter provides a local level of pre-processing or job consolidation before commanding the corresponding application server. The filter could also act on behalf of the application server if it possesses the knowledge required for the requesting member. Each application filter is application-specific. Mobile application developers need to provide their own application filter. Each application filter should be mapped to one mobile application. The group leader receives the encapsulated message from a member. The leader-side group-based messaging component provides the mechanism to decompose the message and forward different portions of the message to the corresponding components and application filters. To differentiate or identify the application filter needed to handle a specific segment of a message, application ID could be used. The corresponding application filter then performs pre-processing task before sending to server. It could act as a command pool in order to collect the commands from members for reducing uplink message to server. For instance, location information can be collected from the leader and the leader reports the group location to the location server. This is our proposed group-based location management scheme. We also proposed a group-based data management application in which the group leader will consolidate the data requests from members for a listening period. A single consolidated query will be sent to the server upon the expiration of the listening period.

Chapter 4

Group Management in Group-based Model

To adopt the group-based model and capitalize on its application, there is a need for system configuration to form an initial set of groups, with respect to the location and the mobility of each mobile host. We propose a dynamic group formation algorithm for clustering a set of mobile hosts with similar mobility together as a group for this purpose in Section 4.1. Afterwards, mobile hosts continue to move around, joining and leaving groups as they move along. In Section 4.2, the basic approach for membership maintenance to handle the situation of mobile hosts to join or leave a group will be discussed. Enhancement in further reducing the local communication overhead in group maintenance is illustrated in Section 4.3.

4.1 Dynamic Group Formation Scheme

To assist groups to be formed, we employ the dynamic group formation algorithm as depicted in Figure 4.1. The algorithm is used for group formation as well as leader election. The algorithm aims at achieving an agreement and a consistent decision in leader election for group formation with as few message passing rounds as possible.

Leader election is governed by a leadership score, s_L. The higher the score, the more potential is a certain mobile host to become a leader. Leader election proceeds by looking for the host with a highest leadership score in a vicinity. The leadership score of a mobile host m with a set of neighbors $\mathcal{N}$ is defined by the equation:

$$s_L = w_1 \sum_{j \in \mathcal{N}} s_{m,j} + w_2 |\mathcal{N}|,$$

where w_1 and w_2 are weights to the two factors, namely, the degree of affinity, $s_{m,j}$, and the connectivity, $|\mathcal{N}|$, and $w_1 + w_2 = 1$. Recalling that degree of affinity reflects how similar the mobility is between two mobile domains in terms of their distance and movement factors (see Section 3.1.2), the higher the degree of affinity to a mobile host with its neighbor, the higher potential is the mobile host to become a leader of the group. The higher the connectivity, the more beneficial is for the host to join the group of the potential leader. As there is no group being formed in the period of system configuration, the only information that can be used for making decision on leader election is its neighbors' location information and its own connectivity obtained through collaboration between itself and its neighbors. After groups are formed, each mobile host stores the information about its group so that the leader election criteria is changed to see whether the host itself is potential to be the next primary leader on behalf of the group. Thus, another score calculation method is required in the execution of the system and this leads to the introduction of *secondary leadership score* in Chapter 5.

The dynamic group formation algorithm consists of two phases. In the first phase, the mobility similarity score of each neighbor of a mobile host m is determined according to the location and velocity provided by the neighbors. The leadership score is then calculated locally and broadcast to its neighbors. In the second phase, the mobile hosts will choose the neighboring host with the highest leadership score to join, by sending it a JOIN message. The mobile host which receives the JOIN message will become the leader of the group. After the algorithm is completed, the mobile hosts are clustered into groups.

Group Formation Algorithm:

1. Each mobile host m broadcasts a HELLO message appended with its own location, velocity and transmission range to its neighbors.
2. When m receives the Hello message from a neighbor n, it will check the the distance between both hosts according to the predicted location after time parameter τ. If the hosts are within range, n will be added to m's neighbor list and $s_{m,n}$ is calculated.
3. Host m records the connectivity degree when HELLO messages are received. It sums up the degree of affinity ($s_m = \sum_j s_{m,j}$) with respect to its neighbors and calculate the leadership score, s_L.
4. If m does not possess long-range communication capability to the location server, $s_L = 0$.
5. After s_L of each host m is calculated, m broadcasts the score to its neighbors in the neighbor list.
6. Host m compares all obtained leadership scores and joins to the host l with the highest score by sending l a JOIN message. Host ID is used to break tie in leadership score comparison, in case it is necessary.
7. If any host m receives a JOIN message, it will become the leader and add the neighbor n to the member list. Upon elected, m will not send any JOIN message to other host k although k may have a larger leadership score than m.
8. It is possible that a host m sends a JOIN message to another host k before receiving the JOIN message from another host. If that is the case, m will send a LEAVE message to k. Host k then removes m from its member list.

Figure 4.1: Group formation algorithm.

Similar to [11], a weight-based scheme is adopted for electing a leader for a group. However, the distribution mechanism of weighted scores in [11] is time consuming because the mechanism requires information exchange throughout the whole network, i.e., the time required depends on the size of the network. It is not desirable in dynamic mobile ad hoc network. In our algorithm, only one message passing round of leadership score distribution is required and another message passing round of JOIN message is used for a cluster to stabilize. Mobility similarity comparison is also not considered in [11]. Only the mean value of the absolute "relative" speed, i.e., the average of absolute velocity magnitude difference between a host and the neighboring hosts over a time interval, is used for mobility similarity comparison in [4]. In our work, the movement directions of mobile hosts are also taken into consideration for group formation.

Figure 4.2 depicts an example for the execution of the dynamic group formation algorithm. At the time of system activation, in order to obtain the knowledge about the surroundings of each mobile host, each mobile host broadcasts a HELLO message to

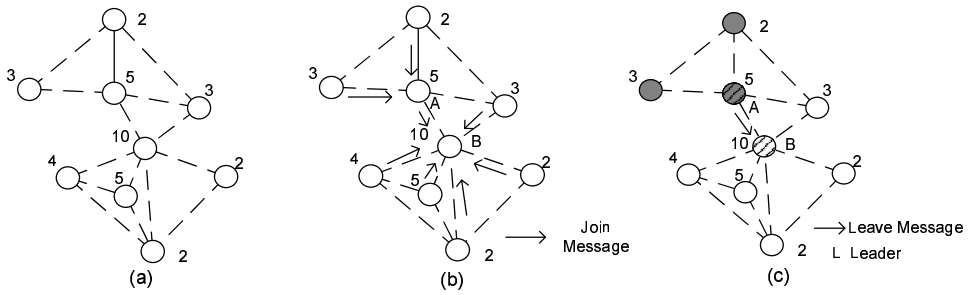


Figure 4.2: An example to illustrate the dynamic group formation algorithm.

its neighbors. As depicted in Figure 4.2(a), every mobile host then calculates its own leadership score, s_L, by using location information piggybacked in HELLO messages from its neighbors. The leadership score of each mobile host is then passed to its neighbors, as depicted in Figure 4.2(b). Now, there is enough information for a mobile host to make a decision for joining its potential leader which has the highest leadership score among its neighbors. A mobile host examines the leadership scores from its neighbors and joins to its potential leader by sending a JOIN message. However, it is possible that a mobile host joins to a potential leader but this potential leader has already joined to another mobile host. As depicted in Figure 4.2(b), host **A** receives JOIN messages from its neighbors after it joined to mobile host **B**. Thus, it is required to have a stabilization mechanism for avoiding the situation. To stabilize the group formation, any mobile host m who previously joined to another mobile host is required to send a LEAVE message to the previously joined leader, if this host m receives JOIN message from its neighbor. As shown in Figure 4.2(c), host **A** who has previously joined to host **B** needs to send a LEAVE message to this previously joined leader, after it received a JOIN message from other mobile hosts for stabilization of the group formation. At last, two groups are formed in the example in which host **A** is the leader of one of the groups and host **B** is the leader of another group.

Consider another scenario that three mobile hosts, say a, b and c, are going to form groups by the execution of group formation algorithm. After exchanging the information about leadership score, a decides to join b while b desires to join c. b will send a LEAVE message to c for stabilization if b has sent a JOIN message to

c previously and now receives a JOIN message from a. However, it is not possible that c is going to join a because c must have the highest leadership score among these three mobile hosts, taking into account the hosts' ID as tie-breaker in case of identical leadership score. c will not send any join message to b or a. As a result, no "cyclic" join situation can possibly occur. Finally, after the group formation algorithm is executed, two groups would be formed, where there is a group with a and b and another singleton group with c.

4.2 Membership Maintenance

Membership maintenance involves the mechanisms for a group leader to handle the joining and leaving of a mobile host to and from a group. In group joining, a mobile host, which is known as a *group seeker*, has a will to find the most suitable group to join. The most straightforward way is to ask the surrounding mobile hosts about the group information available. We term this simple approach as basic join procedure. Similarly, the member of a group leaving a group should notify the leader about its will to leave. The detail about the leave procedure can be found in Section 4.2.2.

4.2.1 Basic Join Procedure

The basic join procedure is depicted in Figure 4.3. In the basic join procedure, a mobile host m that wants to join a group must first request for the group information from its neighbors, by broadcasting FIND_GROUP messages. A neighbor receiving the message will reply a GROUP_INFO message with group information pertaining to its existing group. Group information received from the neighbors includes: leader's host ID, predicted group location and predicted group velocity according to the time of the latest updated location and movement information, and the range $\boldsymbol{r}$ of the group.

The *group finding process* in the basic join procedure requires m to select the most

suitable group available for joining. There may be several potential groups available for joining, according to the received group information from the neighbors. The distance between each potential group G and m itself, $dist(m, G)$, and the distance between the predicted location of group G and the predicted location of m itself after time period τ, $dist(m_\tau, G_\tau)$, are determined. If both $dist(m, G)$ and $dist(m_\tau, G_\tau)$ are smaller than communication range $\boldsymbol{r}$, G becomes a candidate group for joining. If there are several candidate groups, the degree of affinity, $s_{m,G}$, between each candidate group G and m is calculated. Mobile host m will then join the candidate group G with the highest degree of affinity, by sending JOIN message to the leader of the group along with its ID, location and velocity for recalculating new group location and movement information. The leader of the selected group also inserts a new member record for the joining host to its member list. In contrast, there may be the absence of potential group for joining. In this case, m will create a new group and it becomes the leader of this singleton group.

Basic Join Procedure:

1. The mobile host (*group seeker*) m obtains all the related group information by broadcasting a FIND_GROUP message to its neighbors.
2. A neighbor that receives the FIND_GROUP message replies with a GROUP_INFO message to m with the group information to which the neighbor belongs.
3. Group seeker m receives information from neighbors about groups they belong to.
4. For each group G discovered, add G to the set of potential groups if $dist(m, G) < \boldsymbol{r}$ and $dist(m_\tau, G_\tau) < \boldsymbol{r}$.
5. if the set of potential groups is not NULL then
   send a JOIN message to the leader of the group G^* with the highest degree of affinity, S_{m,G^*}
   store the group leader and group location information and movement information
6. else // no group is available for this moment
   create a new group
   set itself to be the leader

   end if

Figure 4.3: Basic join procedure.

4.2.2 Leave Procedure

A mobile host m may leave a group, either voluntarily or involuntarily. Voluntary leave may occur with user power-off, hand-off, moving out-of-range of the host with respect

to its group and so on. Involuntary leave occurs with sudden system or communication failure or temporary disconnection of the mobile host due to signal distortion or signal loss. In this thesis, we will present the voluntary leave case. Timeout mechanisms can be adopted to tackle involuntary disconnection. A form of heart-beat messages can be generated from mobile hosts to the leader periodically to "renew" their status for being group members, in case involuntary departure due to failure is to be handled. If the leader notices that there is no renewal of a membership after timeout, it assumes the departure of this member and removes the record from the member list. Since the member's departure affect the group location and velocity, the leader is responsible for recalculating the group location, according to the member's latest updated location information stored. In voluntary leave, as depicted in Figure 4.4, the leaving member signals its departure by sending a LEAVE message to the leader when it detects that its distance from the group leader is going to exceed the threshold in the near future. The leader will remove the member from the member list after the LEAVE message is received. The leader also recalculates the group location and velocity according to the latest update location and velocity of this leaving member.

Leave Protocol for mobile host:

1. Send LEAVE message to leader.
2. Join another group by executing the join protocol.
3. If no group is available, create own group by itself.

Leave Protocol for leader:

When leader receives the "leave" message from a host,

1. Remove member record from the member list.
2. Group location and velocity is recalculated.

Figure 4.4: Host interaction for a host leaving a group.

As the cost to maintain a strong consistency at any moment of time would be very high in a dynamic mobile computing environment, providing an exact solution for maintaining a strong consistency will be an overkill to the system performance. Thus, weak consistency is preferred occasionally in a mobile computing environment which means that the system may be in an inconsistent state for a short period of time. For example, it is possible that at a certain moment, there could be leaders of

two groups maintaining the record of a single mobile host (a member "belonging" to two groups in the viewpoint of leaders). This happens when a member of one group leaves its existing group by sending a LEAVE message to the group leader. Before the group leader receives the LEAVE message, the group finding process of the departed member finds another new group to join. The new group leader receives the JOIN message before the old group leader receives the LEAVE message. If this is the case, two group leaders maintain the member record of the same mobile host for a short time period while the member will only know about the newly joined group leader. This inconsistent state would not last for long because the leader of the old group will eventually be aware of the departure of the mobile host, by receiving the LEAVE message from the mobile host after intra-group location update is triggered or the timeout of the departed member's membership. The system will return to a consistent state eventually and there is only little impact on the mobile applications. In group-based location management scheme, there is only a very slight impact on the group's location information for GBL in the short inconsistent state since uncertainty is already one of the inherent characteristics in location management. For group-based data management, it could be said that there is no impact since every member, which only knows a single group leader, will send its queries to the group leader it knows.

4.3 Reducing Local Communication Overhead in Group Maintenance

Owing to the dynamic movement of mobile hosts, a number of mobile hosts will occasionally leave their existing group and find other new group to join. These changes in group membership dictate a group maintenance mechanism. In particular, it is important to provide operations for the group leader or clusterhead to handle the leave and join events for a mobile host. Thus, group management is an integral component in mobile ad hoc network clustering and in our group-based model. In Section 4.2,

we proposed a basic join procedure to handle the group joining from a wandering mobile host. To handle the join event, there is a need for a mobile host to select a group to join, before the processing of the join event. The join event can be handled in a straightforward manner when the joining host reports its presence to the group leader.

The more interesting step lies in the selection of an appropriate group and its leader for a wandering mobile host. This is referred to as the *group finding process*, and the mobile host looking for a group to join as the *group seeker*. Besides the performance of the group-based location reporting scheme when groups are formed, it is also useful to study the overhead incurred in group management, with different strategies in the group finding process. According to our preliminary study in Section 4.3.1, there could be a high local communication overhead in the group management mechanism, due significantly to the messages generated in the group finding process.

In this section, the techniques for reducing local communication overhead in the group finding process are investigated. Two new methods are proposed for a mobile host to locate for an appropriate joining group. The detail of these two new *join procedures* can be found in Section 4.3.2.

4.3.1 Communication Overhead in Membership Maintenance

Group management involves the mechanisms for maintaining the membership of mobile hosts in the groups, including protocols for handling the join and leave events of mobile hosts with respect to a group. In the join procedure, a mobile host (group seeker) needs to find an appropriate group by executing the *group finding process*, so that the impact on the stability of the target group can be minimized. In addition, a join event will be generated to inform the leader of the joining group for the exchange of essential information between the group seeker and the leader. A group seeker needs to interact with neighbors from other groups in order to gather group information and make a group joining decision. In Section 4.2.1, the *basic* join procedure is described,

in which the group seeker makes inquiry to potential neighbors for admission into various groups simultaneously. Based on the simulation study on the basic scheme, it is discovered that there would be quite a high local communication overhead involving the group seeker and its neighbors. Analysis into the cause leads us to the design of two improvements on the group finding process in the join procedure, as described in subsequent subsections.

Preliminary Study

Figure 4.5 depicts the results of a preliminary performance study on the group management with the basic join procedure, with varying mobile host population. The detail setting of the experiment could be found in Section 6.3. In our preliminary study, both the total number of local messages and the number of GROUP_INFO messages increase drastically with the number of mobile hosts in the mobile environment. However, the rate of increase in the number of FIND_GROUP messages is very small when compared with the number of GROUP_INFO messages. In addition, the simulation also reveals that GROUP_INFO messages contribute to a significant portion of all the local messages.

In the basic approach, the hosts which are the neighbors of the group seeker will report their group information back after they receive the FIND_GROUP message. As the mobile host population increases, the number of neighbors surrounding the group seeker increases. Thus, many reply messages containing information about the relevant groups are produced for a single FIND_GROUP message originated from a single group seeker. Among the replied GROUP_INFO messages, a significant portion of them, i.e., those originated from the neighbors belonging to the same group, are basically identical. As a result, unnecessary GROUP_INFO messages will be generated from the neighbors. This situation should be rectified.

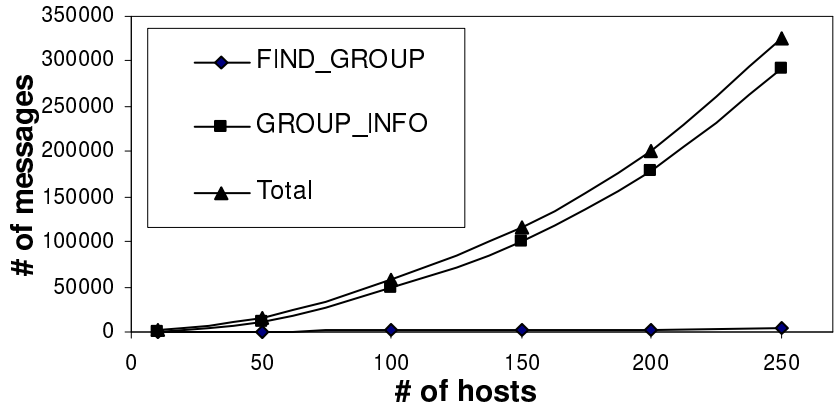


Figure 4.5: Communication overhead in basic join procedure.

4.3.2 Improvement in Join Procedure

Based on the observation in Section 4.3.1, two improved join procedures are proposed. The first improved join procedure restricts the set of mobile hosts that are eligible to replying the group information back to the *group seeker*. In particular, only a group leader is eligible to replying with the FIND_GROUP message to a group seeker, as shown in Figure 4.6. With this constraint, duplicated GROUP_INFO messages containing information about the same group will be eliminated, since there is only one representative answering the request of the group seeker. This is called the *leader-only join procedure*, since only the leader will reply the FIND_GROUP message with the appropriate information. When a group seeker looks for a suitable group to join, it will broadcast a FIND_GROUP to its neighbors. In the leader-only join procedure, the mobile host that replies a GROUP_INFO messages back to the group seeker after it received the FIND_GROUP message must be a group leader. Then the group seeker makes its own judgement on which one of available groups is the most suitable to join. The detail of the execution of the procedure can be found in Figure 4.6.

Although with leader-only join procedure, duplicated messages from the same group are eliminated, we can observe that the group seeker will only join one group in the end. To further reduce the number of reply messages, a leader that receives the FIND_GROUP message can make a judicial decision to reply or not, by evaluating the degree of affinity, $s_{m,G}$, between the potential group G and the group seeker m. If the

Leader-only Join Procedure:

1. The mobile host (*group seeker*) m obtains all the related group information by broadcasting a FIND_GROUP message to its neighbors.
2. A neighbor who is a group leader that receives the FIND_GROUP will reply with a GROUP_INFO message to m with the relevant group information.
   A non-leader neighbor will simply discard the FIND_GROUP message.
3. Group seeker m receives information from neighbors about groups they belong to.
4. For each group G discovered, add G to the set of potential groups if $dist(m, G) < \boldsymbol{r}$ and $dist(m_\tau, G_\tau) < \boldsymbol{r}$.
5. if the set of potential groups is not NULL then
   send a JOIN message to the leader of the group G^* with the highest degree of affinity, s_{m,G^*}
   store the group leader and group location information and movement information
6. else // no group is available for this moment
   create a new group
   set itself to be the leader
   end if

Figure 4.6: Leader-only join procedure.

degree of affinity is larger than a predefined *filtering threshold*, θ, the leader will reply the group information back to the group seeker. Otherwise, the request message will be ignored. As a result, groups with low degree of affinity are further filtered out, since they would unlikely be selected in the end. This constitutes our second proposed join procedure, termed *leader-filter join procedure*. As illustrated in Figure 4.7, a group seeker first obtains the information of surrounding groups available from its neighbors by sending FIND_GROUP messages to them. Every neighbor that receives the message will reply a GROUP_INFO message, on which group information corresponding to the neighbor that it belongs to is piggybacked based on two conditions. The first condition is that the neighbor is a group leader while the second condition is that the degree of affinity between the neighbor's belonging group and the group seeker has to be larger than a filtering threshold, θ. If both conditions are satisfied, the neighbor will reply the group seeker with a GROUP_INFO message. After the group seeker receives the replies from its neighbors, it will make a decision about which group to join.

In short, this chapter discusses about the group management issues involved in realizing the group-based model and the group-based framework. These issues include group formation and membership maintenance. A preliminary study on the basic group

Leader-filter Join Procedure:

1. The mobile host (*group seeker*) m obtains all the related group information by broadcasting a FIND_GROUP message to its neighbors.
2. A neighbor n who is a leader of group G that receives the FIND_GROUP will further perform a filtering step:
   if the degree of affinity, $s_{m,G}$, is greater than the filtering threshold θ, then
   reply m with a GROUP_INFO message with the relevant group information
   A leader neighbor failing the filtering condition or a non-leader neighbor will simply discard the FIND_GROUP message.
3. Group seeker m receives information from neighbors about groups they belong to.
4. For each group G discovered, add G to the set of potential groups if $dist(m, G) < \boldsymbol{r}$ and $dist(m_\tau, G_\tau) < \boldsymbol{r}$.
5. if the set of potential groups is not NULL then
   send a JOIN message to the leader of the group G^* with the highest degree of affinity, s_{m,G^*}
   store the group leader and group location information and movement information
6. else // no group is available for this moment
   create a new group
   set itself to be the leader
   end if

Figure 4.7: Leader-filter join procedure.

joining in membership maintenance is conducted. It shows that the group finding process in the join procedure contributes to a signification portion of local communication overhead. Improvement of the join procedure is proposed for reducing the local messages involved in group finding. Simulation study is conducted and will be discussed in detail in Section 6.3.2. Another critical component in group management is leadership maintenance, which will be discussed in detail in next chapter.

Chapter 5

Leadership Maintenance

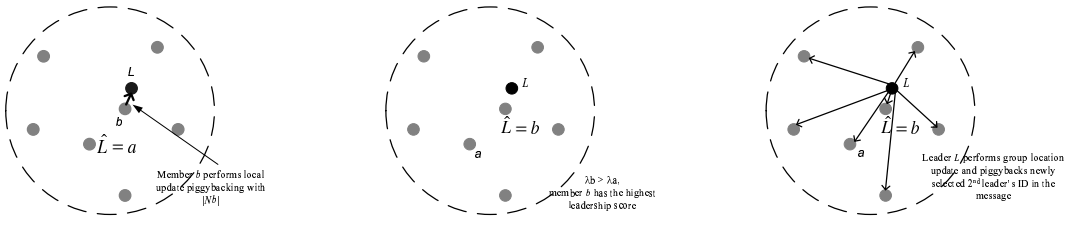
Due to mobility of group members, a group leader may be decoupled from its group voluntarily or involuntarily. Leadership change among group members is unavoidable; it is important to develop an efficient scheme to deal with the issue. An intuitive approach in addressing the change of leadership is to re-execute a mobile ad hoc network clustering algorithm (either in a demand-driven or periodic manner). However, it is not a suitable approach in the group-based location management context because it could be costly in terms of communication cost if the clustering algorithm is re-executed, especially in large-sized groups. Message volume generated due to host interaction could be large. Additional uplink messages will be introduced for informing the server about the newly formed groups. The performance is also degraded because a leader is absent before the completion of the clustering algorithm. If the group leader fails, the location updating messages from members to the leader are wasted and members need to execute the leader election algorithm instead. During the election, the location server is not able to receive any location update message about the members in the group, hence degrading system performance.

In order to maintain dynamic leadership within a group, we propose a leadership maintenance scheme with the aid of a secondary leader in the group. There are three major components in our leadership maintenance scheme, built on top of our group-

based model and the intra-group location management component (see Section 3.2). The first component is a *secondary leader determination* strategy, a strategy to determine who will be a potential secondary leader to take over the group leader's role when necessary. The second component, termed *turnover activation policy*, is a policy for identifying an endangered primary leader and making the proper decision on when to trigger the *turnover procedure*. The third component *turnover procedure* is triggered to really hand over the primary leader's duty to the secondary leader. This component consists of procedures for handling the turnover and notifying members about the change of primary leader. In general, there should always be a potential secondary leader in a group ready for assuming the leadership from the primary leader. We assume that there are occasional but infrequent message loss, as exhibited by most practical systems.

5.1 Secondary Leader Determination

To select a secondary leader responsible for taking over the activities of a primary leader after the primary leaves, the existing primary needs to gather the information required for secondary leader determination amongst members during system execution. Two pieces of information, namely, *member-neighbor connectivity*, $|N_m|$, and *degree of affinity*, $s_{m,G}$, between a member m and a group G, are maintained. Member-neighbor connectivity of m is the number of *member-neighbors* of m. The member-neighbors, N_m, of m are those neighbors of m belonging to G. The procedure can be illustrated with an example as shown in Figure 5.1.



(a) Member b performs local update to leader L

(b) Leader L calculates λ_b for member b

(c) Group location update needed, with new leader piggybacked

Figure 5.1: Secondary leader determination.

The member-neighbor connectivity information will be piggybacked on local location update message, as depicted in Figure 5.1(a). The primary leader L obtains the degree of affinity between a member m and its group G by using the local location update information from m. The secondary leadership score, λ_m, of m is defined as $\lambda_m = \omega_1 s_{m,G} + \omega_2 |N_m|$, where ω_1 and ω_2 are weights to the two factors, degree of affinity and member-neighbor connectivity, with $\omega_1 + \omega_2 = 1$. Whenever primary L receives a local location update message from a member m, the secondary leadership score of m, λ_m, is calculated and stored, as exemplified in Figure 5.1(b). If a mobile host is not able to communicate with the remote server, its member-neighbor connectivity will be set to negative infinity, making it ineligible for being appointed as a secondary leader. At the moment when a group location update is generated, the member with the highest secondary leadership score is selected as the secondary leader $\hat{L}$, whose ID will be piggybacked on the group location update message, as in Figure 5.1(c). Due to dynamic host movement, the secondary leader $\hat{L}$ may leave the group. When $\hat{L}$ leaves the group, L removes the record of $\hat{L}$ from the member list. The next highest ranked member is then chosen to be the new secondary leader. L will also generate an intra-group location update to all members with information about the new secondary leader. Thus, members are kept informed of changes in secondary leader as soon as possible. This is beneficial in involuntary leader changing situation, which a group leader departs from its group involuntarily, since the possibility of handing over the primary leader's job to a departed secondary leader is reduced. For intra-group location update, L will send a group location update message only to each member, but not to the server, to reduce expensive uplink traffic. There is no change in the information stored in the leader about the latest updated location, velocity and update time. In each group location update message, the ID of the current secondary leader $\hat{L}$ is piggybacked. Primary leader L will also keep track of its own secondary leadership score, λ_L, to monitor for the possibility that secondary leader's score surpasses its own, thereby triggering the turnover procedure.

To determine the member-neighbor connectivity in group G, each member m main-

tains a *member-neighbor list*, storing the list of *member-neighbors* and *member-neighbor relationship expiry time* (or simply *expiry time*) of each member-neighbor. The member-neighbor relationship of a member to a neighbor is considered valid before its expiry time. The validity assumption serves as a tradeoff for the accuracy of member-neighbor information against the number of status update messages in the presence of host mobility. Thus, a soft state technique is adopted for collecting the connectivity information. To maintain the member-neighbor relationship between member-neighbors, each host m broadcasts an expiry time renewal message before the expiry time, together with the member's leader ID and the new *valid duration* calculated adaptively. The new valid duration is the estimated time required for m to travel from its current location to the boundary of the group, according to the relative speed between the velocities of m and G. Upon receiving the renewal message, m will examine its leader ID and the leader ID in the renewal message. If they are the same, both hosts belong to the same group; m will update the expiry time of that neighbor, adding the neighbor into the list with the expiry time if it is a new neighbor. We adopt a lazy approach for the expired member in the member-neighbor list. The expired member-neighbors in the list will be removed lazily only when the connectivity of the member is to be determined.

Although more accurate member-neighbor list can be maintained with a conservative approach in member-neighbor relationship renewal, this may induce high number of local messages. To improve system performance, a *relaxed* member-neighbor renewal strategy is employed to reduce the local communication overhead. The new valid duration is computed as the time required to travel from the current location to the boundary of the group *plus* its transmission range distance. To further reduce the number of local messages, *piggybacking* technique is adopted. Whenever there is a group location update or an intra-group location update, the primary leader calculates a new valid duration and embeds it in the group location update messages. Members of the group then update the group location information and renew the member-neighbor relationship of the primary with the message.

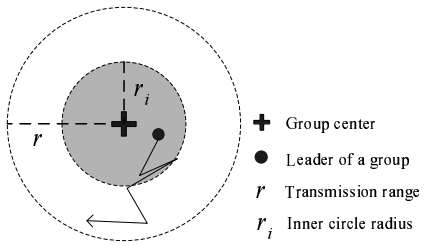


Figure 5.2: Group-based model with the notion of inner circle.

5.2 Turnover Activation Policy

Turnover activation policy is a mechanism for determining *when* the leader is required to turn over its current job to the secondary leader, that is, the primary has a high tendency to stay near the margin of a group and leave its group sooner or later, thus termed a *tend-to-leave* leader. Rather than waiting for a *tend-to-leave* leader leaving the group and reacting on demand, the policy identifies this kind of endangered leader proactively and triggers the turnover procedure. The reason is that it is desirable to have a leader often staying near the group center, rather than staying near the group margin. If it is discovered that the leader is often far away from the group center, it is better to hand over its job to another potential secondary leader, by invoking the turnover procedure.

Each group leader executes the *turnover activation policy* periodically. Figure 5.2 depicts the model employed in the turnover activation policy. An *inner circle* is introduced with radius r_i, centered at the group center. We define a *stayIndex* to indicate the tendency of a leader staying within the inner circle. The higher the value of the index, the higher tendency that the leader stays within the inner circle. The *stayIndex* is measured in each location sampling; it is incremented by one if the leader L is within the inner circle; otherwise, it will be divided by a *drop factor*, χ. The *drop factor* is adjusted adaptively according to the difference between the degree of affinity of the leader to the group in the previous and the current sampling periods, i.e., $s_{L,G}^{Prev}$ and $s_{L,G}$. *Drop factor* will be decreased when there is an increase in the current degree of affinity and vice versa, as given by $\chi^{new} = \chi^{old}(1 - (s_{L,G} - s_{L,G}^{Prev}))$, where two thresholds, χ_{min} and χ_{max}, are predefined to bound the drop factor, with $\chi_{max} > \chi_{min} > 1$.

Figure 5.3 depicts the turnover activation policy. In short, the leader L executes the turnover activation policy in each location sampling period. *Drop factor* and *stayIndex* are evaluated. There is a predefined *turnover threshold*, *TurnThr* ($0 < TurnThr < 1$). If $stayIndex < TurnThr$ and the secondary leadership score $\lambda_L < \lambda_{\hat{L}}$, L is treated as a tend-to-leave leader and the turnover procedure will be invoked. Note that there is no need for the turnover activation policy for a group of size two, since the turnover will not achieve its desired goal for such a group.

Initial Condition:
1. stayIndex = 1

StayIndex Adjustment at L:
1. $\chi \leftarrow \chi(1 - (s_{L,G} - s_{L,G}^{Prev}))$
2. if leader stays in the inner circle then
3. $\quad stayIndex \leftarrow stayIndex + 1$
4. else
5. $\quad stayIndex \leftarrow stayIndex/\chi$
6. end if

Turnover Activation Policy at L:
1. for each location sampling period do
2. $\quad$ adjust $stayIndex$
3. $\quad$ if ($stayIndex < TurnThr$ and $\lambda_L < \lambda_{\hat{L}}$) then
4. $\quad\quad$ turn over the job to $\hat{L}$ by invoking the voluntary turnover procedure
5. $\quad\quad$ revert L back to an ordinary member
6. $\quad$ end if
7. end for

Figure 5.3: Turnover activation policy.

5.3 Turnover Procedure

Turnover should occur when the leader leaves the group (on demand), or when it is too risky to rely on the leader which has a strong tendency to stay near the margin of the group (anticipatory). There are three possible situations when a primary leader L is required to turn over its role. First, the turnover activation policy may identify L as a tend-to-leave leader. Second, voluntary leave event occurs when L moves out of the group range $\boldsymbol{r}$, or L intends to power down or to disconnect itself. These two situations are handled by the *voluntary turnover procedure*. Finally, involuntary leave event occurs when there is a sudden failure of L. The procedure in handling this situation is termed *involuntary leader changing procedure* (i.e., *involuntary turnover procedure*).

5.3.1 Voluntary Turnover Procedure

Figure 5.4 depicts the host interaction involved in the voluntary turnover procedure. To begin with, the existing primary leader L sends a TURNOVER message to the secondary leader $\hat{L}$. The TURNOVER message contains the group member list, the group location and velocity, and the latest update time, latest updated group location and velocity. If L initiates a voluntary leave, it will remove itself from the group member list before sending TURNOVER. After $\hat{L}$ receives TURNOVER, it considers itself the new primary leader and constructs a member list according to the message received. The new primary then identifies a new secondary based on its member-neighbor list, computes its valid duration, and broadcasts a CHANGE_LEADER message to all its members, with message content of previous leader ID, new secondary leader ID and its own new valid duration. After members receive CHANGE_LEADER, they update their own record about the new primary and secondary leader, and the expiry time of the new primary.

Owing to asynchrony of message passing and host mobility, a member m^* which has already departed group G for group G^* may receive a CHANGE_LEADER message from the new primary leader $\hat{L}$ of G. It occurs when the old primary leader L of G does not receive the LEAVE message from m^* before the voluntary turnover procedure is invoked by L. The new primary $\hat{L}$ would still consider this departed member m^* as a group member according to the member list it received and would send a CHANGE_LEADER message to m^*. When m^* receives CHANGE_LEADER, it should reply back a LEAVE message to $\hat{L}$ to remove itself from the member list of the latter, i.e., group G.

Owing to potential message loss, it may happen that a subset of members fail to receive the CHANGE_LEADER message. Upon timeout, those hosts would execute the *involuntary leader changing procedure* when they discover the loss of the existing primary and are unaware of the secondary becoming the new primary. This is a simplification to reuse an existing protocol in a slightly different but yet applicable context.

Voluntary Turnover Procedure:

1. Primary leader L unicasts message TURNOVER to secondary leader $\hat{L}$, along with group information.
2. Secondary $\hat{L}$ that receives TURNOVER converts itself into the new primary leader and identifies an appropriate new secondary $\tilde{L}$ from its member-neighbor list.
   $\hat{L}$ broadcasts a CHANGE_LEADER message, with L's ID, $\tilde{L}$'s ID and $\hat{L}$'s new valid duration to all other members in the member list.
3. Member m that receives CHANGE_LEADER compares its current primary leader ID with the old leader ID in the message.
   If both IDs are equal, things are alright. Member m changes its primary leader to be the message sender. It also updates its record of the secondary leader and the new primary's expiry time, according to the information embedded in the message.
   Otherwise, m has changed group, a LEAVE message will be sent to the message sender to finish off with the group switching.

Figure 5.4: Voluntary turnover procedure.

Details in handling such a scenario will be discussed next.

5.3.2 Involuntary Leader Changing Procedure

In the involuntary leader changing procedure, each member utilizes the member-neighbor list and auxiliary information to detect the involuntary departure of the primary leader. We adopt a lazy approach in determining the necessity of involuntary leader turnover upon timeout. In other words, we only initiate the involuntary leader changing procedure when there is a need for a member to report its location via a local location update message. Involuntary leader changing is initiated with a *leader changing update* (CHANGE_UPDATE) message, on which a regular local location update message is piggybacked. The detail of the involuntary leader changing procedure is shown in Figure 5.5.

Before a member m issues a local location update, the expiry time of the primary leader L is checked. If it is valid, L is probably still around and a regular local location update message is sent to L. If it has expired, L may have disappeared from the view of m, which will then initiate the involuntary leader changing procedure, by sending a CHANGE_UPDATE message to its secondary leader $\hat{L}$. The latest group location update time stored in m and the primary leader L's ID are included in the message. The latest group location update time indicates the timestamp of leader L when L last issued a group location update to members.

When a member m initiates involuntary leader changing procedure to its secondary

Involuntary Leader Changing Procedure:

1. When a local update is to be issued by host m and the expiry time of primary L is up, m sends a CHANGE_UPDATE message to its secondary $\hat{L}$ with the latest group update time and its current leader ID.
2. Host h receives a CHANGE_UPDATE message from m.
   a. case h is a primary leader:
      // h has already become a primary leader triggered by other members but m does not know this
      h replies m with a CHANGE_LEADER message containing h's previous primary leader ID, current secondary leader ID and h's remaining valid duration
   b. case h is a secondary leader:
      if (h's leader ID == m's leader ID) then // m and h belong to the same group
         if leader L's expiry time is not up yet then // no change in leader
            h sends back a REJECT message to m
         else
            h probes L with an IS_ALIVE message
            if h receives before timeout a YES message from L then // no change in leader
               h sends back a REJECT message to m // primary leader is still healthy
            else // timeout and h becomes the new primary leader
               h broadcasts a CHANGE_LEADER message with L's ID only
   c. case host h is a member:
      if (h's leader ID == m's leader ID) then // m and h belong to the same group
         if leader L's expiry time is not up yet then // no change in leader
            h sends back a REJECT message to m // primary is healthy
         else if (latest group update time in $h >$ latest group update time in CHANGE_UPDATE) then
            h sends another CHANGE_UPDATE message to h's secondary leader $\hat{L}$ and waits for reply
            if a REJECT message is received before timeout then // primary is healthy
               h sends back REJECT to m // propagate REJECT
            // else timeout, h cannot contact new leader, so it leaves the group
         else
            h probes L with an IS_ALIVE message
            if h receives before timeout a YES message from L then
               h sends back a REJECT message to m // primary leader is still healthy
            else // timeout and h becomes the new primary leader
               h broadcasts a CHANGE_LEADER message with L's ID only
3. Member m waits for the reply.
      if REJECT is received before timeout then // primary is healthy
         m sends location update to existing primary leader L
      else // timeout, m no longer belongs to the group
         m tries to find another group to join
4. Host n receives a CHANGE_LEADER message from host l.
      if (n's existing primary leader L's ID == previous leader ID in CHANGE_LEADER) then
         // change is ready for installation
         if CHANGE_LEADER contains new secondary's ID and primary's new expiry time then
            // (case in paragraph 2a)
            change primary leader to l
            update secondary leader and primary leader's expiry time
         else // (case in paragraph 2b or 2c)
            change primary leader to l
            send a regular local location update to l

Figure 5.5: Involuntary leader changing procedure.

leader by a CHANGE_UPDATE message, the receiving secondary will re-confirm the existence of the primary leader by sending a probe (IS_ALIVE) message to check whether the primary has left the group. If the primary is still around, the involuntary leader changing procedure terminates with a REJECT message back to m, which updates its information. If the leader has departed from the group, the secondary declares itself as the new primary and broadcasts a CHANGE_LEADER message with the previous primary leader ID. This is slightly different from the CHANGE_LEADER message generated in voluntary turnover, without new secondary leader and primary leader's new expiry time information, since the new primary leader has only limited

knowledge about the group and members within the group. Members of the group receiving CHANGE_LEADER will verify whether the change is legitimate, updating the new primary leader information, followed by a regular location update to the new primary.

In the presence of message loss, some members may not receive an expiry time renewal message from their leader. If this is the case, the primary leader's expiry time that they store will eventually expire. Upon expiry and when member m needs to report a local location update, the involuntary leader changing procedure is invoked. However, the primary leader still exists within the group and it is not necessary to change the leadership. To address this problem, the secondary leader that receives the CHANGE_UPDATE message will check the leader's expiry time. If it has not yet expired, the primary is still considered to be healthy and the secondary receiving the CHANGE_UPDATE message will reply with a REJECT message so that m will discontinue the leader changing procedure. It is also possible that the secondary leader missed the latest expiry time renewal messages from the primary leader, but the primary is still around. To guard against this, the secondary will confirm with the primary for its departure with a probe IS_ALIVE even if the expiry time is up. As with above, if there is no response, the secondary assumes the new leadership and broadcasts CHANGE_LEADER for the leadership turnover.

After the voluntary turnover procedure is finished, there may be some members failing to receive the CHANGE_LEADER message from the new primary leader. Before the expiration of leader's expiry time, those members may issue normal local location update messages to the old leader. The old leader will simply discard the messages, even if it is still in the group. Thus this old leader behaves as if it were not in the group, thereby unifying the failure mode as observed by those negligent members. The old leader's expiry time will eventually be up at those negligent members, who will then send CHANGE_UPDATE messages to their secondary leader. At this moment, the secondary leader has already become the new primary leader. So, when this new primary receives a CHANGE_UPDATE message, it will reply a CHANGE_LEADER

message to those negligent members only through unicast message passing. In this case, the group information is available and the CHANGE_LEADER message will contain information of the selected secondary leader ID and the primary leader's remaining valid duration.

Besides missing CHANGE_LEADER message from a new primary, there is a scenario that a member m misses a group location update message which indicates the change of secondary leadership. When a CHANGE_UPDATE message is issued by m, it is sent to its preferred secondary leader p, asking p to take over. Now p may be the secondary leader $\hat{L}$ or an old secondary leader. If p is an old secondary leader, it may still be a member of the group or may have become a member of another group with a different primary leader. Upon receiving a CHANGE_UPDATE message from m, p compares its current leader ID and the leader ID embedded in the local location update message. If p is still a member of the existing group (i.e., both IDs are the same), p will check for the expiration of primary leader's expiry time. If it has not yet expired, primary leader may still be valid and a REJECT message is replied to m. Otherwise, the latest group update time values stored in p and in the local update message are compared. If the update time stored in p is larger than that stored in the local update message, it will propagate the CHANGE_UPDATE message to the secondary leader p' that p prefers. Host p' will then execute the involuntary leader changing procedure. If the update time in p is less than that in the local update message, indicating that p missed a notification of secondary leader change, p then considers itself to be the new primary and broadcasts CHANGE_LEADER.

5.4 Departure of Secondary Leader

Since a secondary leader is also a member in a group, it is possible that it will depart the group voluntarily or involuntarily. As mentioned in Section 5.1, if the departure of secondary leader is voluntary, it will send a LEAVE message to the primary leader.

The primary that receives the message will remove the membership of this departed secondary leader. Then the primary will select the next highest ranked member to be the new secondary leader and issue an intra-group location update to the members to inform them of the newly selected secondary leader.

If the departure of secondary leader is involuntary, the leader will eventually be aware of the departure of the secondary leader with the expiration of the valid duration of the member-neighbor relationship for the departed secondary leader. The primary removes the record of this departed secondary leader from the member list. If it is the case, the leader will select a new secondary leader and inform the members when it is the time for an intra-group location update. For the worst case that both primary leader and secondary leader leave the group involuntarily, the members in the group will eventually issue a leader changing location update to their potential secondary. Since there is no response from the departed potential secondary, all members will leave the group (group breakup) and attempt to join to other groups.

Chapter 6

Group-based Location Management Scheme

Built upon our group-based model, a group-based location management scheme (GBL) is developed. There are three types of location update. In the first two types, it is about the location management within a group. Thus, both types contribute to the integral part of the group-based framework while the third type is an application of the framework to provide location reporting service to the server in a group-based manner. The first type is called *local update*, is the mechanism for reporting location information to a group leader from its members. The strategy is termed as *local location updating strategy*. *Intra-group location update* as the second type is about the mechanism for a leader reporting the group location information back to its members. These two types of location update form the local level of group-based location management. The third type is about report location information from a group leader to the server residing in a fixed network, which is termed as *group location update*. This type contributes to the global level of group-based location management. Intra-group location update and group location update basically follow the same location updating strategy, termed as *group location updating strategy*, except that there may be a need for extra intra-group location updating because of leadership maintenance (see Chapter 5).

6.1 Local Location Updating Strategy

In local location updating strategy, a group member periodically samples its current location and velocity. Such information will then be compared with the predicted location according to its latest updated location and velocity. The deviation, in terms of the distance between the predicted location and the current location, will be measured. If the deviation is larger than a prescribed threshold D, an update message will be sent to the leader. In the update message, the current location and the velocity of the member are attached. The information about the group location and velocity will be recalculated according to the location information of the member in the update message. The threshold value that will trigger an update is determined by the *degree of affinity* between a mobile host group member m and its group G. The next threshold value D is determined by $D = r \times (1 - e^{-s_{m,G}})$ [32], where $s_{m,G}$ is the degree of affinity between mobile host m and its group G (see Section 3.1.2). The higher the similarity value, the higher the threshold value will be.

6.2 Group Location Updating Strategy

In group location updating strategy, it provides strategy for both intra-group location update and group location update. The leader measures and monitors the deviation of the group from the prediction and reports to the location server when the deviation exceeds another prescribed threshold T_G. There are three types of events affecting the group location and the velocity: join event, leave event and local location update event from group members. A group leader receives the relevant location information from its members. The group location and velocity will be recalculated. The plain dead-reckoning (*pdr*) approach [56] is used for making decision of location reporting to the location server in group location update and to its members in intra-group location update. A distance deviation threshold T_G is predefined at the beginning of system execution. This threshold T_G may be used throughout the system execution, and it

Table 6.1: System characteristics in the experiments.

Parameter	Default value
Region area	$100m \times 100m$
Transmission range, r	$30m$
Minimum speed, s_{min}	$0.1ms^{-1}$
Maximum speed, s_{max}	$0.5ms^{-1}$
$s_{j,k}$ weights: α, β	0.5, 0.5
s_L weights: w_1, w_2	0.5, 0.5
λ_L weights: ω_1, ω_2	0.5, 0.5
Time parameter, τ	$5s$
Location sampling period, τ_s	$2s$

may also be adjusted dynamically. If the distance between the current location and predicted location of a group is larger than T_G, a location update message will be sent from the leader to the location server and members. In general, more sophisticated dead-reckoning approach, such as adaptive dead-reckoning (*adr*) [56], can be applied to group location updating. Information sent from a leader to the location server includes the group location, velocity, and the member list of the group.

In the case of singleton group, no intra-group update will be performed. Group update to server is still required and individual-based plain dead-reckoning (*pdr*) will be applied, that is, the leader compares its current location with the predicted location from the latest location information updated to the server; if the deviation is larger than the threshold T_G, the leader will send a location update message to the server.

6.3 Performance Study

A simulation model is built for studying the performance of GBL. The simulation is implemented by using CSIM18 [45]. For simplicity, but without loss of generality, it is assumed that all mobile hosts possess the long-range communication ability. In other words, they can communicate with the location server and are eligible to be a leader. Location positioning system such as GPS is built in at each mobile host. It is

also assumed that disconnection is uncommon in the experimental ad hoc networks. All mobile hosts move freely according to the random waypoint (without pause) [29] movement model within a specific region of $100m$ by $100m$. Each host moves with a minimum speed, s_{min} = $0.1ms^{-1}$, and a maximum speed, s_{max} = $5ms^{-1}$. In Sections 6.3.1 and 6.3.2, we allow the leader to move freely only when it is not too far away from the center and constrain its movement when it moves towards the margin of a group, for simplicity. This ensures that the leader will not move out of the group it is serving, and hence dictating a change of leadership. We will relax this constraint when studying the performance effect of the proposed leadership maintenance scheme in Section 6.3.3. Every host can interact with each other within the transmission range of ***r***=$30m$. Both group location update threshold T_G and individual location update threshold D_i are set to $20m$. Our experimental setting is summarized in Table 6.1. Each host will sample its current location every $\tau_s = 2$ seconds with its location positioning device and decide whether to report the location according to the location updating scheme. Each experiment is simulated for 600 seconds.

6.3.1 Performance of GBL

In this section, we consider the primary performance metrics, namely, the number of update messages generated to the location server to keep track of the locations of all the mobile hosts. In particular, we compare GBL with individual-based plain dead-reckoning (*pdr*) location update method, which is shown to yield satisfactory performance. We conduct four sets of experiments to study the performance along four different perspectives, including varying mobile host density, varying time parameter τ (see Section 3.1 and 4.2), and different weight settings in degree of affinity and in leadership score. Detailed parameter setting for the four experiments is depicted in Table 6.2.

Figure 6.1 illustrates the performance of our first set of experiments, namely, conventional individual-based location update scheme versus GBL. It can be observed that

Table 6.2: Parameter settings of the four experiments.

Parameter	Experiments			
	Mobile host density	Time parameter τ	Degree of affinity weights, $s_{j,k}$	Leadership score weights, s_L
Number of hosts M	10-250	100	100	100
$s_{j,k}$ weights: α, β	0.5, 0.5	0.5, 0.5	α: 0.1-0.9	0.5, 0.5
s_L weights: w_1, w_2	0.5, 0.5	0.5, 0.5	0.5, 0.5	w_1: 0.1-0.9
Time parameter τ	5	0 - 40	0 - 40	0 - 40

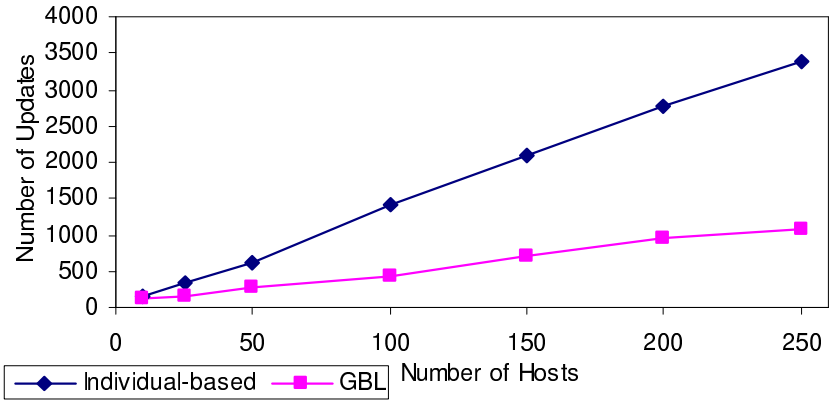


Figure 6.1: Performance study on GBL with different mobile host densities.

GBL is very effective in reducing the number of update messages in medium to high mobile host density environments when compared with conventional individual-based location updating. With high mobile host density, the group size will be large and the group-based scheme is able to consolidate the large number of update messages from within a group for reporting purpose. However, if the mobile host density is very low, each group only contains few members and each single join or leave event of a mobile host to or from a group becomes significant in affecting the group's location and velocity, which implies the decreasing of group stability. This would also result in the cascading departure effect, in which a single departure leads to a series of departures by other members in the group. The group location will be dramatically changed.

Our second set of experiments compares the effect of the time parameter τ in GBL. As shown in Figure 6.2, GBL does not seem to perform well with large value of τ. This is because large value of τ will lead to a decrease in group size, since there will be fewer mobile hosts satisfying the group formation criteria to stay within the group

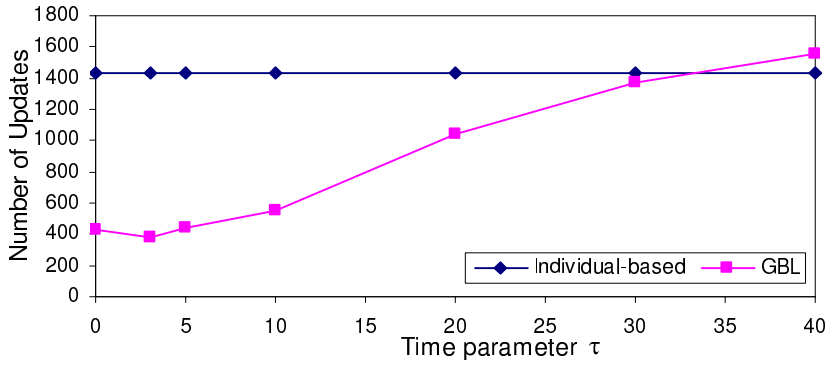


Figure 6.2: Performance effect of GBL with time parameters τ.

boundary with a longer period of time. With smaller group size, there will be more groups and the benefit brought about by GBL for each group will diminish and the penalty inset by the instability of the small groups increases that it gradually becomes out-performed by individual-based scheme. With a very small value of τ, the location of a host plays a major role when it looks for a group to join; the moving speed and direction are less significant. Hosts with very short group staying period will likely join the group, thereby leading to a slight decrease in the degree of affinity between members in the group. This in turn leads to a slight increase to the group instability and a slight performance degradation.

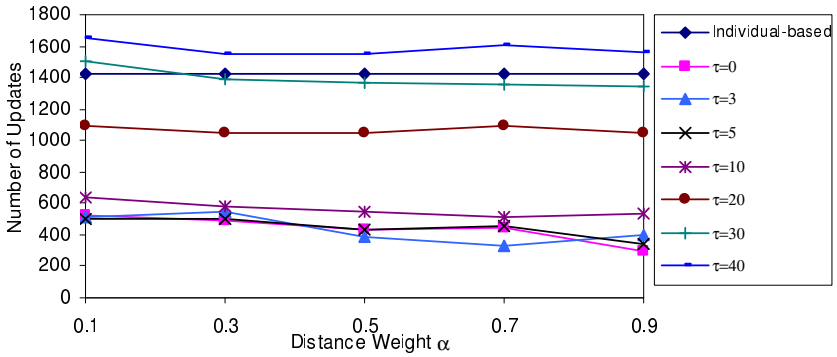


Figure 6.3: Weights in degree of affinity.

Our third set of experiments investigates into the performance effect of different weights of degree of affinity, as depicted in Figure 6.3. It can be observed that the impact of different weight settings in the calculation of degree of affinity is not quite significant. However, there is still a general trend that the number of updates decreases as α increases, i.e., the distance factor is more significant than the movement factor

in the degree of affinity. With a small α, a mobile host may rather join a group **A** with much similarity in velocity strength and direction but in the margin of the group than another group **B** with a smaller distance difference but relatively less similar in velocity strength and direction. This situation increases the group instability and leads to a slight increase in the number of update messages to the location server. In addition, it is found that GBL is more sensitive to the weight when τ is small, since there will be more groups and each mobile host has a higher degree of freedom in selecting an appropriate group. The change in weights could lead to a choice of a different group to join. Finally, the general trend of decreased performance with increased value of τ is observed, consistent with the observation in the second experiment.

Finally, the effect of the weights in leadership score calculation is studied. From the results in Figure 6.4, the varying weights do not seem to lead to a major performance difference. The only observation is that extreme values of w_1 or w_2 could lead to a slight increase in the number of update messages in general, probably due to the strong bias towards one of the two factors. Again, it is consistent that the performance degrades with increasing time parameter τ.

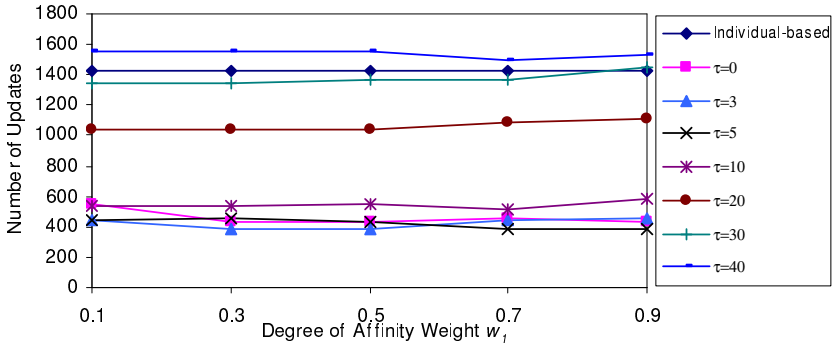


Figure 6.4: Weights in leadership score.

In this section, we have studied the performance effect of GBL scheme in terms of the number of uplink messages for location update. Results show that GBL is effective in reducing the number of location updates to the stationary server. However, to get the benefit of the GBL scheme, the number of local messages for group management would be another concern. As shown in Figure 6.5, a large amount of local messages for group management could be generated, including messages for group formation al-

gorithm, and join (basic join procedure) and leave procedures. The local messages for location management have little impact on the total number of local messages. From our preliminary study, as depicted in Figure 4.5, a significant portion of local messages is due to the group finding process. There is a significant increase in the number of GROUP_INFO messages when the number of hosts in the environment increases. As a result, we proposed two improved join procedures for enhancing the performance of the GBL scheme by reducing the number of local messages for group management (see Section 4.3.2). The number of local messages for group maintenance drastically decreases when improved join procedures are adopted, in turn leading to the decrease of total number of local messages, as depicted in Figure 6.5. In order to study the actual benefit of the GBL scheme, the number of local messages should be interpreted as only one of factors in the aggregated cost which considers both long-range and local communication overheads (see Section 6.3.2). In Section 6.3.2, we study the performance effect of our proposed join procedures (including the basic join procedure) in detail in terms of aggregated cost, and the result shows that the GBL scheme outperforms the individual approach when our proposed improved join procedures are adopted.

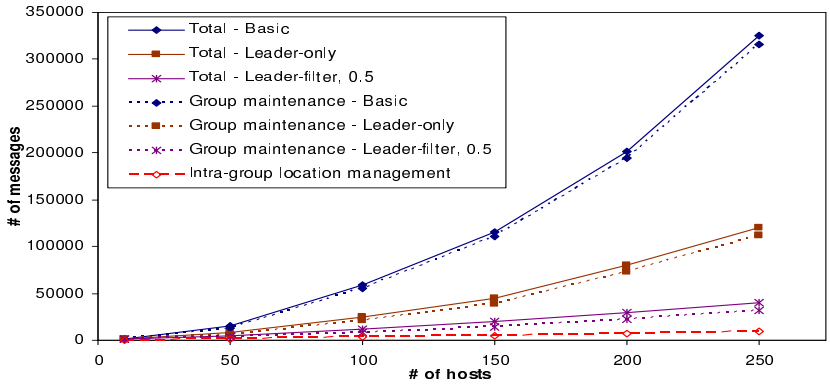


Figure 6.5: Local communication overhead in the GBL scheme.

6.3.2 Performance on GBL with Different Join Procedures

A simulation model is built for studying the performance of GBL regarding the basic and the newly proposed join procedures. The performance of the improved join procedures with different mobile host populations is studied. A total of three different

variants of the join procedure are compared. These include the basic, leader-only and leader-filter join procedures. Thus, the effect of different constraints on group selection can be studied. For the leader-filter join procedure, different filtering threshold values, from θ = 0.1 to θ = 0.9, are studied in the simulation. The results are illustrated in Figure 6.6. It is obvious that there is a drastic performance enhancement for both leader-only and leader-filter join procedures. A highly significant improvement can be attained when the leader-filter join procedure is used, with high filtering threshold values $\theta \geq 0.5$. This reflects that limiting the set of neighbors of a group seeker to participate in the group finding process is an effective way for alleviating the local communication traffic, especially in medium and high population density environments.

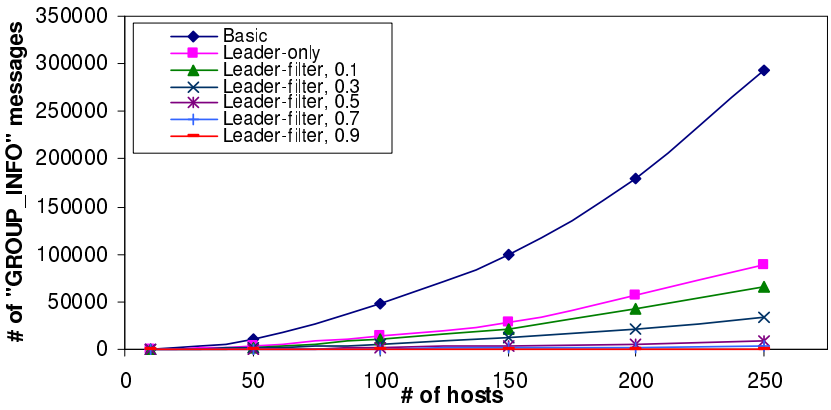


Figure 6.6: Communication overhead with different join procedures.

The impact on the number of group location update messages to the server with different join procedures is investigated, since they involve the use of expensive uplink channel. The restricted choice of joining groups may impact the number of groups formed and hence the number of group location update messages. As in Section 6.3.1, we compare GBL with individual-based plain dead-reckoning (*pdr*) location update method with our join procedures and the results are depicted in Figure 6.7. From the figure, leader-only and leader-filter (with θ =0.1 and 0.3) join procedures create only minimal negative impact on the number of group location update messages. By choosing a high filtering threshold, there is an increase in the number of location update messages to the location server in our GBL scheme. The individual-based approach

outperforms the GBL scheme when $\theta = 0.9$. GBL only begins to outperform the individual-based in high host population environments when $\theta = 0.7$. However, there is only a little reduction in the number of GROUP_INFO messages, as depicted in Figure 6.6. This is because with a high filtering threshold, the number of neighbors eligible to participate in the group finding process will be smaller. There is a lower chance for a group seeker to find a suitable group for joining and the group seeker may then form a singleton group. The higher the value of θ, the more stringent the restriction in the group finding process. Thus, the number of singleton groups increases, which induces an increase in the number of group location update messages to the server.

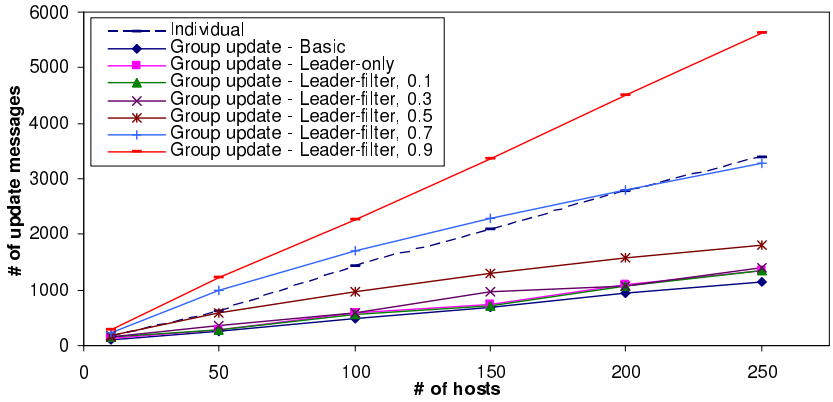


Figure 6.7: Performance of GBL with different join procedures.

From Figure 6.7, it is not difficult to observe that the number of group location update messages increases with the new join procedures (see Section 4.3.2), and this is due to the formation of more singleton groups. To verify this observation, we measure the number of singleton groups at each moment, as well as the total number of groups. The results on the average number of groups are depicted in Figure 6.8. There is a higher number of singleton groups formed with a higher filtering threshold for leader-filter join procedure, and all our new join procedures generate more singleton groups than the basic scheme. This is because the sole members in those singleton groups experience difficulty in finding suitable group for joining because of the tight restriction with a high filtering threshold in the leader-filter join procedure. It is also interesting to note that the total number of groups for the new join procedures even suffers from

a further increase than the number of singleton groups. This is due to the fact that as the chance of forming singleton groups increases as explained in Figure 6.7, these singleton groups may coalesce into groups of size of two. As a result, there are more groups with a smaller group size formed, since there is a higher chance for a singleton group missing the chance to join another non-singleton group. Furthermore, it can be observed from Figure 6.8 that the average group size for leader-filter join procedure with a high filtering threshold ($\theta = 0.5$) grows from 1.32 with 10 mobile hosts to 3.14 with 250 mobile hosts, whereas this size grows from 2.21 to 3.58 for the basic join procedure. Thus, the performance gap between these join procedures narrows with a higher host population.

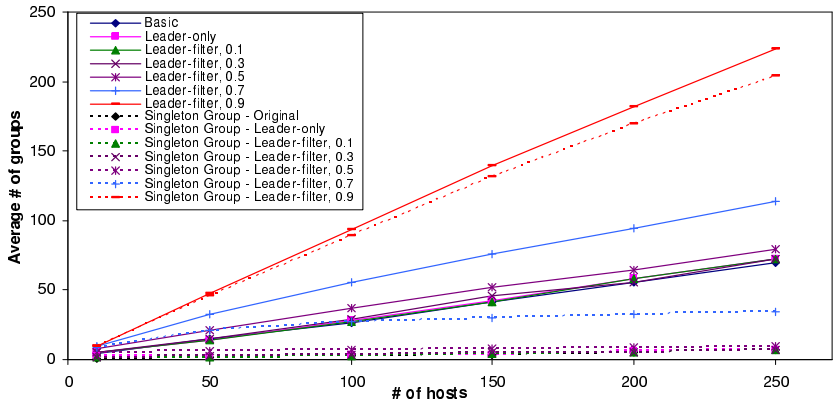


Figure 6.8: Number of groups and singleton groups.

To further evaluate the impact on overhead generated by singleton groups (created due to bad location or aggressive filtering) as compared with those generated by normal group membership change (created due to host movement from one group to another), we measure the number of leave events. The number of leave events will define a lower bound on the total number of group finding messages. The ineffectiveness of group formation, manifested in the form of singleton groups, can be studied with the *find/leave ratio* or *F/L ratio*, which is defined as the ratio of FIND_GROUP message count to leave event count. The higher value this ratio is, the more singleton groups contribute to generating FIND_GROUP messages.

Figure 6.9 illustrates the effect with respect to the ratio with different join procedures. In general, the ratio is high in low population density environment and drops to

a stable level with median to high population density. When the leader-filter join procedure is applied and the filtering threshold value becomes higher, the ratio increases and becomes higher than other join procedures in different host population environments. More singleton groups are created, as shown in Figure 6.8. The significant increase in F/L ratio happens because many mobile hosts are experiencing difficulty in finding a group to join and are forming singleton groups, thereby running the group finding process periodically, often in vain. This is manifested as a low leave count, a high FIND_GROUP message count, and hence a high F/L ratio. In other cases, the probability to find a group with more than one host is relatively higher. The leave count increases and the FIND_GROUP message count decreases so that the F/L ratio becomes smaller even at the same low host population. Even though the F/L ratio in high filtering threshold leader-filter join procedure is higher than other join procedures in low population environments, there is a general trend that the F/L ratio drops and becomes stable in medium to high host population environments. The reason is that the proportion of singleton groups becomes smaller as the mobile host population increases.

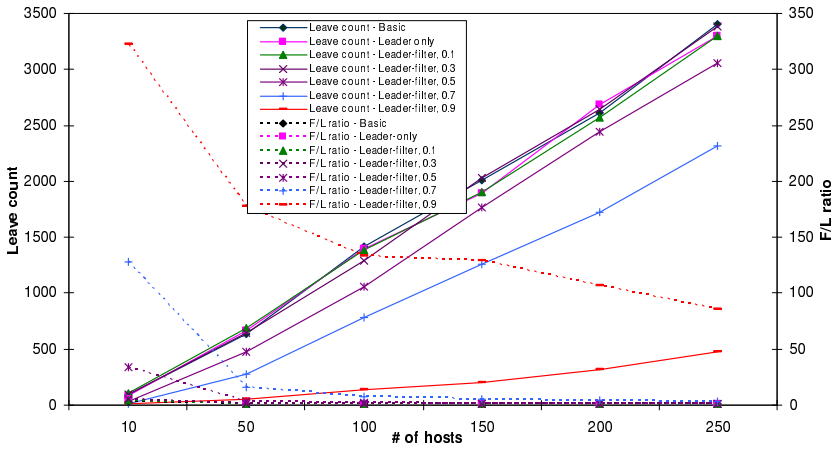


Figure 6.9: Performance of different join procedures with respect to F/L ratio.

To study the performance of the GBL scheme from the view of all mobile hosts, both short-range and long-range communication costs must be taken into account. In particular, we are interested in the consolidated performance for the whole system, through a cost function measuring the aggregated cost, $C = c_s \times n_s + c_l \times n_l$, where

c_s and c_l are the costs of a short-range message interaction in the ad hoc network and the cost of sending a long-range message through the uplink channel respectively, while n_s and n_l are the number of short-range messages and long-range messages respectively. It is assumed that $c_l > c_s$ and the cost in broadcasting a short-range message is the same as the cost in unicasting a short-range message and that the cost of sending a message is constant. We define $\xi = \frac{c_l}{c_s}$ to be the global/local cost ratio. Thus, $C = c_l(\frac{n_s}{\xi} + n_l)$. Without loss of generality, we could assume that c_l is of unit cost, since it reflects the cost of standard individual-based location update scheme. The aggregated cost C at a client population of 250 with varying global/local cost ratio ξ is depicted in Figure 6.10.

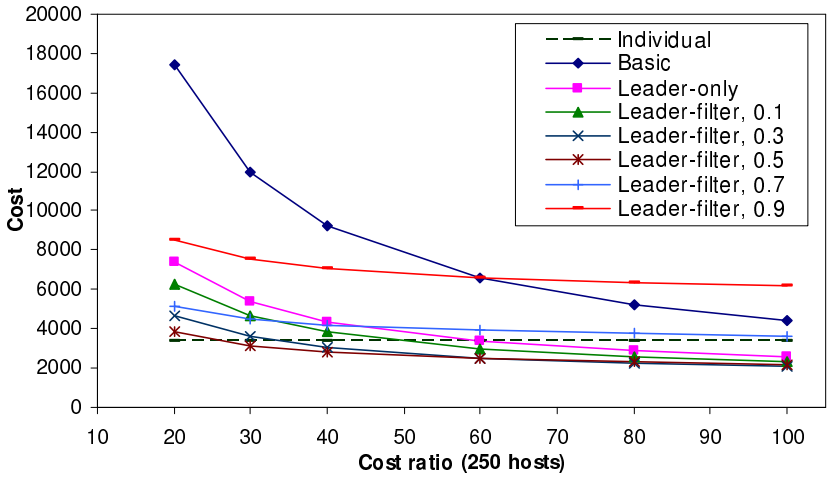


Figure 6.10: Consolidated cost of GBL with different join procedures.

In Figure 6.10, it is obvious that the cost for individual-based scheme remains constant, since it involves no local message. In the basic join procedure, there is a large number of local messages generated, thus resulting in a high aggregated cost for the mobile hosts, though the load at the location server decreases, with a reduction of the number group location update messages to the server. When the leader-only or leader-filter join procedure is applied, the aggregated cost for mobile hosts decreases and the two improved GBL schemes outperform individual-based approach starting from a medium low to high cost ratio respectively. In the leader-filter join procedure, the breakeven point is at a lower cost ratio ξ with a higher filtering threshold, $\theta = 0.5$. However, the individual-based approach outperforms the GBL when $\theta = 0.7$ and $\theta =$

0.9. It is because the increase of the number of singleton groups leads to the increase of local communication overhead by periodic execution of group finding process for those singleton groups. On the other hand, a low value of θ is not effective in doing the filtering. Thus, the value of filtering threshold θ could not be too high or too low, in order to get the most benefit of the leader-filter join procedure. Hereafter, the filtering threshold value will be set to 0.5 as the default value in other experiments, because the GBL scheme yield the best performance when $\theta = 0.5$.

6.3.3 Performance on GBL with Leader Maintenance

We conduct simulation to investigate two different leadership maintenance approaches in addressing the leadership maintenance problem. The first one is a straightforward extension of our clustering algorithm for group formation (see Section 4.1), with re-execution (hereinafter referred to as re-run cluster algorithm). Whenever a leader departs from a group, all members become leaderless members, not belonging to any group. The clustering algorithm will then be invoked to form groups among those leaderless members. The second one is the leadership maintenance scheme as discussed in Chapter 5. In this approach, two different variants in maintaining member-neighbor connectivity are studied. The first variant is a straightforward realization of our leadership maintenance scheme (called the basic scheme). The conservative member-neighbor renewal strategy is adopted and no renewal information is piggy-backed in group location update messages. The second one is an improvement on the leader maintenance scheme, in which a relaxed member-neighbor renewal strategy is adopted and piggybacking technique is employed. This is called the improved scheme (see Section 5.1).

In this section, we remove the assumption that a leader is allowed to move freely only around its group center. In other words, all mobile hosts can really move freely according to the random waypoint movement model [29] within the defined region (see Table 6.1). Other parameter settings are as follows. The drop factor, χ, is initially

set to 2.0. The drop factor bounds are $\chi_{max} = 5.0$ and $\chi_{min} = 1.25$. The inner circle radius r_i is 0.7 times of the range ***r***. The two factors in determining the degree of affinity carry equal weights, i.e., $\alpha = \beta = 0.5$. The secondary leadership score is also computed with equal contributions from the degree of affinity and member-neighbor connectivity, i.e., $\omega_1 = \omega_2 = 0.5$.

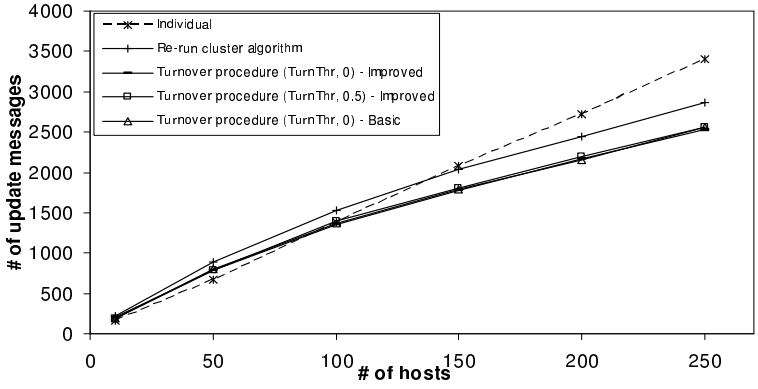


Figure 6.11: Performance of GBL with leadership maintenance.

We studied the impact on the number of expensive uplink group location update messages to server with two different approaches in leadership maintenance, when compared with the individual-based location updating scheme. As depicted in Figure 6.11, the GBL scheme with leader-filter join procedure (see Section 4.3.2), with filtering threshold value $\theta = 0.5$, is experimented in these two approaches. This is because it was shown that it yields a best performance. In Figure 6.11, it is obvious that the GBL scheme with the two different leadership maintenance approaches are effective in reducing the number of group update messages to location server in medium to high host population environments. In particular, our proposed leader maintenance scheme and its variants outperform the individual-based scheme at a high population and they consistently outperform the re-run cluster approach. The reduction of update messages in our scheme stems from the fact that there are more groups formed with the re-run cluster approach, as depicted in Figure 6.12. Thus, more group location update messages are generated for conveying to the server information about the newly formed groups after the clustering algorithm has been executed. There are similar performance effects on the number of group location update messages to the server on

the different variants of our leadership maintenance scheme. We also study the performance effect of the proposed leadership maintenance scheme with the use of the turnover activation policy (*TurnThr* = 0.5) and without the policy (*TurnThr* = 0). As depicted in Figure 6.11, both experiments yield similar results.

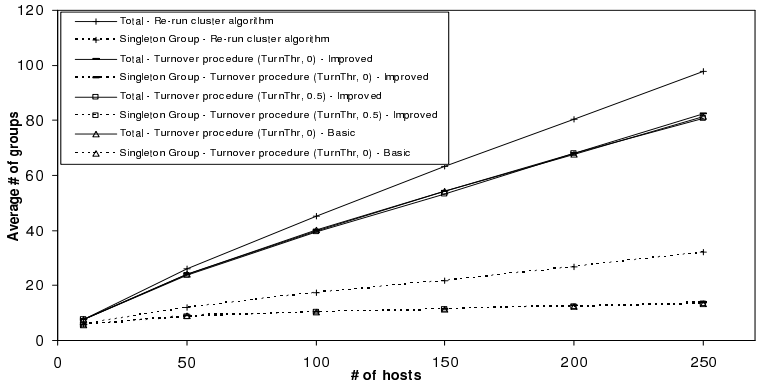


Figure 6.12: Number of groups and singleton groups.

Figure 6.12 shows the results of the average number of groups and average number of singleton groups with different host population densities. The average number of groups and singleton groups decrease with our proposed leader maintenance scheme. In other words, the group size increases with our leadership maintenance scheme. As the group size increases, the impact on the join or leave event from a mobile host is reduced, thus increasing the group stability. The re-run cluster approach induces more groups and more singleton groups because each member in a group suddenly becomes an individual leaderless host after the group leader departs from the group. Meanwhile, fewer group location update messages are generated in our leader maintenance scheme, thereby alleviating the load of the server. Furthermore, with our leadership maintenance scheme, the duration of leader absence can be minimized. The GBL scheme is still functioning properly in the course of leadership changeover, without having to suspend location update activities to server for leadership maintenance through re-clustering. Thus, GBL scheme with our leader maintenance approach has only a small impact on updating location information to the location server since there is less time spent for a primary leader handing over its job to a secondary, when compared with re-executing the clustering algorithm among those leaderless members.

From the result, there is little difference in average number of groups and singleton groups for different variants of our leadership maintenance scheme, except that the improved variant of leadership maintenance scheme with turnover activation policy performs slightly better in terms of average number of groups. Nevertheless, our scheme does exert a positive impact on the group stability. The group stability increases when the turnover activation policy is enabled under high population environments.

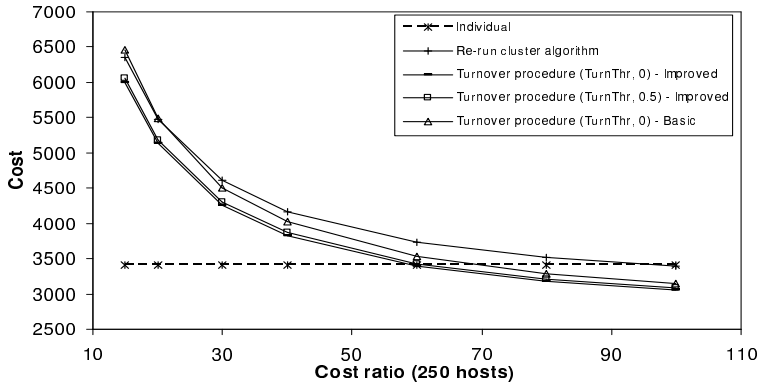


Figure 6.13: Consolidated cost of GBL with leadership maintenance.

To study the performance of the GBL scheme from the collective view of all mobile hosts, both short-range and long-range communication costs should be taken into account. Recalling from Section 6.3.2, an aggregated cost function is defined as $C = c_s \times n_s + c_l \times n_l$, where c_s and c_l are the cost of a short-range message interaction in the ad hoc network and the cost of sending a long-range message through the uplink channel respectively, while n_s and n_l are the number of short-range messages and long-range messages respectively. With global/local cost ratio $\xi = \frac{c_l}{c_s}$, the aggregated cost $C = c_l(\frac{n_s}{\xi} + n_l)$. Without loss of generality, we could assume that c_l is of unit cost, since it reflects the cost of standard individual-based location update scheme. The aggregated cost C at a client population of 250 with varying global/local cost ratio ξ is depicted in Figure 6.13.

From Figure 6.13, it is obvious that the cost for the individual-based scheme remains constant, since it involves no local message. However, the cost of all GBL schemes decreases with increased host population, due to the increasing ease of group formation for the aggregated reporting effect. Thus, all GBL schemes are perform-

ing better than the individual-based scheme at a high cost ratio, but they worsen with lower cost ratio. It is interesting to note that only at the lowest cost ratio will the re-run cluster approach outperform the basic variant of the proposed leadership maintenance scheme, which is already the worst among other variants. This is because more message exchanges between mobile hosts are required in the clustering algorithm and more messages are generated for group location update. In executing clustering algorithm, host interaction is involved in electing a leader for a new group and joining the group by the members. However, in our approach, host interaction is reduced by embedding secondary leader selection information in both local location update and group location update messages, increasing leader's member-neighbor relationship renewal frequency by piggybacking new valid duration in the group location update messages. In addition, fewer message exchanges are required in the turnover procedure. The reduction in the group update message count to the server also contributes in reducing the aggregated cost. As a result, the straightforward approach of re-running clustering algorithm is not that effective, though it is still better than the individual-based scheme at high host population.

In comparison among the different variants of our leadership maintenance scheme, it can be observed that the basic variant is consistently the worst and the improved variant with turnover activation policy disabled performs marginally better than the improved variant with turnover activation policy enabled. The technique of piggybacking renewal messages in group location update messages and relaxing member-neighbor strategy does produce positive effect to our leadership maintenance scheme. However, there is a slight increase in aggregated cost when the turnover activation policy is enabled, though there is also a slight increase in group stability as depicted in Figure 6.12. This is a tradeoff to be considered between the group stability and the aggregated cost.

Chapter 7

Group-based Data Management Scheme

Conventionally, querying data is accomplished by a mobile host issuing a point-to-point query to the server independently. It can be treated as *individual-based approach*, as mentioned in Chapter 1. Each mobile host does not have knowledge about the queries issued from the mobile hosts nearby. Similar query requests would be generated, which can be treated as "almost" duplicated messages. There is a high chance for the mobile hosts in vicinity to generate data queries for similar data items, especially in location-dependent applications. For example, different mobile hosts might well ask for the nearest restaurants with different price ranges in a consumer guidance mobile application. To alleviate this problem, group-based paradigm could be applied and this leads us to the design of the group-based data management (GBD) scheme. That means a set of query requests can be collected within a group and those requests would be consolidated before sending to the databases server. As a result, the number of expensive uplink query messages to the server could be further reduced. In the example above, those queries about nearest restaurants could be consolidated to a single query message within a group before being sent to the stationary server. In automotive traffic information system, drivers nearby often query about similar traffic information, e.g.

drivers may ask for traffic reports in the areas farther away from their current location along the road, such as $100m$ to $250m$ for a driver and $200m$ to $400m$ for another driver. These queries that are within a group could be consolidated as a single query to stationary server for the required information by using the GBD scheme.

As the goal of group-based data management scheme is to reduce the uplink frequency in querying data, there are two major issues that are important for the effectiveness of the scheme, in addition to fundamental issues such as group formation and group management in group-based paradigm. The first issue is the query consolidation procedure, which aims at consolidating members' query requests for reducing uplink traffic. A straightforward approach is that the leader concatenates all the queries received from members over a period of time and sends a single message to the server. However, this results in a large-size message for the query message because the final consolidated query message can be very long, by appending each query to the message. An effective query consolidation procedure should be provided. The second issue is the query listening period, which is the time period for a group leader waiting for the queries from members before the leader sends a consolidated query message to the server. It could not be too long or too short. If it is too long, a member has to wait for longer to get back the answer of its issued query; in turn the query response time increases. If it is too short, more uplink messages will be sent to the server and long-range communication cost will increase. This leads to a tradeoff between the query listening period and the query response time, in turn a tradeoff between the querying cost and the query response time. To realize our group-based data management scheme, we propose a query consolidation procedure and a dynamic query listening period calculation method for addressing these two issues.

7.1 Group-based Data Management Model

In this section, we are going to formulate the system model for the group-based data management scheme. Mobile hosts in the environment form groups under our proposed group-based model (see Chapter 3). Group management and leadership maintenance strategies for group-based mobile environment could be found in Chapters 4 and 5. Semi-hybrid data accessing mechanism is assumed. In other words, hot data items are disseminated by data broadcasting. When there is a query from a user that cannot be answered by the data items from broadcast, a point-to-point query request is triggered to remote server according to the GBD scheme. Remote server then schedules the requested cold data items to be disseminated to all hosts over another broadcast downlink channel.

As shown in Figure 7.1, mobile hosts are clustered into a set of groups, by executing group formation algorithm at system activation. When there are data requests from members in a group that could not be answered from the data broadcast, they issue queries to their group leader for query consolidation. Those queries sent from the members to the group leader are termed as *member queries*. Leader will wait for a time period, termed as *query listening period*, for collecting and consolidating member queries. After the query listening period, the leader sends a consolidated data request to the servers. The answer of the request will then be replied through another broadcast channel. Both the leader and the members that have generated member queries will listen to the data broadcast. Indexing in data broadcast is assumed for reducing the time a mobile host spent listening in active mode for energy saving. The interested parties could check for the data index to be broadcast in the broadcast channel and turn to active mode when the required data item arrives.

The group-based data management scheme is based on relational data model. We consider an n-ary base relation, R, in the database. Range queries will be issued from mobile hosts for selecting entire tuples from the database satisfying the conditioning attribute in each query and its range. For example, a mobile host may want to

know about the hotel information with the price range between HK$500 and HK$1000, but unwilling to spend a four-digit amount. The host will generate a SQL-like query from the mobile application as: select * from Hotel where price $\geq$ 500 and price $<$ 1000. In query processing, each SQL-like query will be formulated as another query expression: $\langle\, relation, attribute, \{beginInter, min, max, endInter\}\,\rangle$, where $relation$ and $attribute$ specify the relation in the database to be requested and the selection attribute of the query respectively. The range of the query is bounded by the min and max values ($min \leq max$), and $beginInter$ and $endInter$ indicate the inclusive equality or exclusive equality of the min value and the max value respectively. If the min value of a query is inclusive, $beginInter$ is set to '[' in the query expression. Otherwise, $beginInter$ is set to '(' in the expression. Similarly, if the max value of a query is inclusive, $endInter$ is ']' in the query expression. Otherwise, $endInter$ is ')' in the expression. Note that if $min = max$, the query is meaningful only when both equalities are inclusive. The query expression mentioned above is a logical representation of a query. When a query is being transmitted, a more compact physical representation is possible. A bit is sufficient for $beginInter$ and $endInter$ to indicate whether the min and max values in a query are inclusive or not. Recalling from our example, the query expression for the SQL-like query will be transformed as $\langle Hotel, price, \{[, 500, 1000,)\}\rangle$. There are two functions that return the $beginInter$ and $endInter$ values of a query. The $beginInter(\cdot)$ function that takes a query as the parameter returns the value either '(' or '[' according to the $beginInter$ value of the query. The $endInter(\cdot)$ function that takes a query as the parameter returns the value either ')' or ']' according to the $endInter$ value of the query. Enhancement in supporting projection and other query operations will be considered in our future work.

7.2 Group-based Data Management Scheme

To request data in the GBD scheme, the data requestor will first send the query to its group leader by a local message, instead of a high cost uplink message to the server.

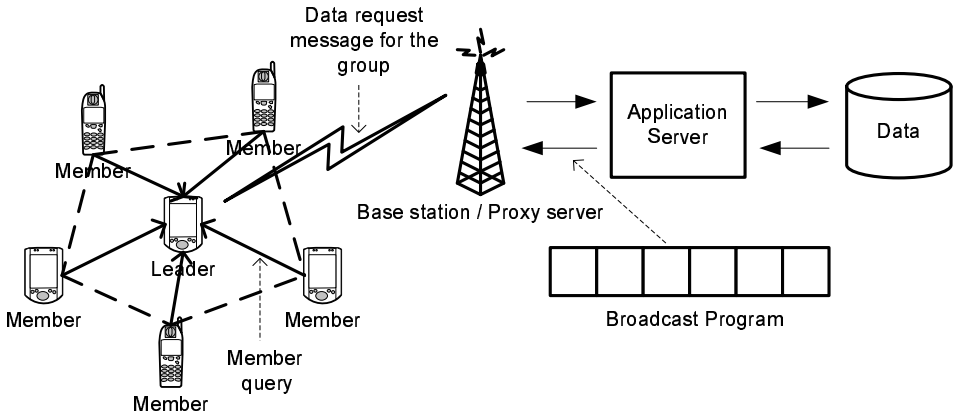


Figure 7.1: The GBD system model.

A group leader stores two separate query lists in the GBD scheme: *consolidated query list* and *outstanding query list*, as depicted in Figure 7.2. A query list contains a set of non-overlapping queries and each query in the list is termed as a *query element*. The *consolidated query list*, $\hat{l}_c$, stores the query elements that are collected and consolidated from the member queries. Those consolidated query list elements are generated according to the query consolidation procedure. The group leader will record the consolidated queries already sent but not yet answered by the server into a query list, termed as *outstanding query list*, $\hat{l}_u$. The wanted data items in the *outstanding query list* are expected to arrive in the next data broadcast. After the group leader receives a member query from the requestor, it will perform the query consolidation procedure, which consolidates the received member query with the existing *consolidated query list*, to generate a new *consolidated query list*. New expiry time will be adjusted according to the new query listening period re-evaluated. This new query listening period is calculated adaptively according to the coverage of the outstanding query list to the consolidated query list. The detail about query listening period adjustment can be found in Section 7.2.2.

If the query listening period is up, a *supplementary query list*, $\hat{l}_s$, will be generated by the group leader. A supplementary query list is a query list containing a set of queries that is extracted from the consolidated query list. Each query element in the supplementary query list is one of the query segments of a query element in the consolidated query list, which is the portion of the query element that could not be covered

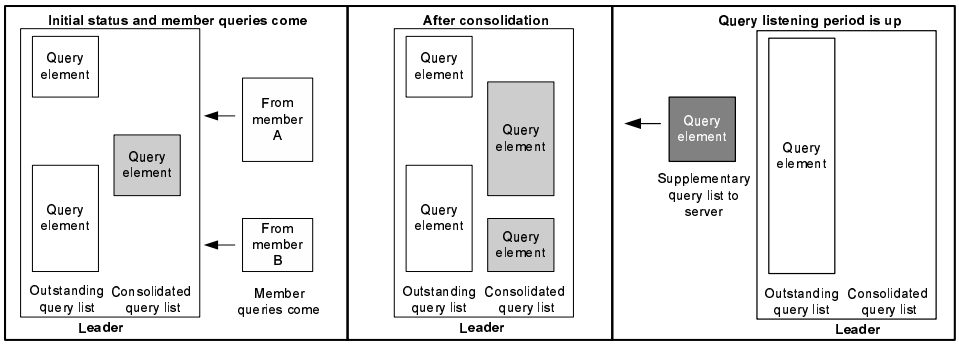


Figure 7.2: Data requesting procedure.

from the query elements in the outstanding query list, as depicted in Figure 7.2. The supplementary query list will then be sent to the server when the query listening period expires. After the supplementary query list is sent, the consolidated query list will be consolidated with the existing outstanding query list to form a new outstanding query list by the $merge(\hat{l}_c, \hat{l}_u)$ function (depicted in Figure 7.4). The function takes two query lists as parameters and is responsible for merging two query lists into a single query list. In the execution of the merge function, an attempt is made for each query element q_a in query list, $\hat{l}_c$, to be consolidated with every query element q_b in another query list, $\hat{l}_u$, by applying the procedure $consolidate(q_a, \hat{l}_u)$ (see Section 7.2.1 and Figure 7.5). The detail about the data querying procedure involved in the GBD scheme is depicted in Figure 7.3. On the other hand, when the leader receives data items from the server, the outstanding query list will be updated, as discussed in Section 7.2.3. The procedure depicted in Figure 7.3 is only executed within those non-singleton groups. For a singleton group that involves only a single member, i.e., the sole leader, in the group, the queries generated by the leader will be sent to the server directly. No query consolidation is required.

7.2.1 Query Consolidation Procedure

In query consolidation, as depicted in Figure 7.5, the query consolidation module tries to consolidate the member query with each query element in the consolidated query list $\hat{l}_c$. The operation for consolidating two queries q_a and q_b is denoted as $q_a \oplus q_b$, where

Procedure $data\_query(q_m)$**:**

// executed when leader receives q_m

1. if $\hat{l}_c = \emptyset$ then

   set *consolidation start time* (see Section 7.2.3) to the current time and timing starts

   end if
2. Consolidate q_m with the consolidated query list $\hat{l}_c$, by invoking $consolidate(q_m, \hat{l}_c)$ (see Section 7.2.1)
3. Adjust the query listening period, according to the query listening period adjustment policy (see Section 7.2.2)
4. if timeout (according to the new query listening period and the start time of query consolidation) then

   send the supplementary query list containing all the segments in $\hat{l}_c$ that is not overlapped with $\hat{l}_u$ to server

   $\hat{l}_u \leftarrow merge(\hat{l}_c, \hat{l}_u)$

   clear the consolidated query list, $\hat{l}_c \leftarrow \emptyset$

   end if

Figure 7.3: Data querying procedure.

Function $merge(\hat{l}_a, \hat{l}_b)$**:**

1. $\hat{l}_{temp} \leftarrow \hat{l}_b$
2. for each query element q_e in $\hat{l}_a$ do

   $consolidate(q_e, \hat{l}_{temp})$

   end for
3. return $\hat{l}_{temp}$

Figure 7.4: Merge function for query lists.

operator $\oplus$ is a binary operator, which takes two queries as operands. The output after the operation is a single query, say q_r, with selection attribute on a single interval. As depicted in Figure 7.6, in total there will be 4 possible scenarios in the relationship between two queries involved in the operation, which leads to different q_r. To simplify our discussion of different cases to be considered in consolidating two queries, let us define the ordering on '(', '[', ')', ']' so that '[' < '(', ')' < ']', '[' = '[', '(' = '(', ')' = ')' and ']' = ']', as defined in Definitions 1 and 2. Based on these two definitions, comparators $\leq$ and $\not<$ can be defined intuitively (see Definition 3).

Definition 1 *For two queries q_a and q_b, $beginInter(q_a) < beginInter(q_b)$ iff $beginInter(q_a) =$ '[' $\wedge$ $beginInter(q_b) =$ '('. Similarly, $endInter(q_a) < endInter(q_b)$ iff $endInter(q_a) =$ ')' $\wedge$ $endInter(q_b) =$ ']'.*

Definition 2 *For two queries q_a and q_b, $beginInter(q_a) = beginInter(q_b)$ iff $(beginInter(q_a) =$ '(' $\wedge$ $beginInter(q_b) =$ '(') $\vee$ $(beginInter(q_a) =$ '[' $\wedge$ $beginInter(q_a) =$ '['). Simi-*

Query consolidation procedure for a query q_m consolidated into a query list $\hat{l}$

Input parameter: query q_m

Input/output parameter: query list $\hat{l}$

Procedure $consolidate(q_m, \hat{l})$**:**

1. $\hat{l}_{temp} \leftarrow \emptyset$
2. $q_c \leftarrow q_m$
3. for each query element q_e in $\hat{l}$ do
   - if ($max(q_c) < min(q_e)$) or
     - ($max(q_c) = min(q_e)$ and $endInter(q_c) =$ ')' and $beginInter(q_e) =$ '(') then
   - // early termination ($\hat{l}$ is sorted in ascending order according to max value of each query element)
     - break
   - end if
   - if $q_c \cap q_e \neq \emptyset$ then
     - $q_c \leftarrow q_c \oplus q_e$
   - else if ($min(q_c) = max(q_e)$) and not($beginInter(q_c) =$ '(' and $endInter(q_e) =$ ')') then
   - // two queries have no intersection but q_e is consecutive to q_c
     - $q_c \leftarrow q_c \oplus q_e$
   - else if ($max(q_c) = min(q_e)$) and not($beginInter(q_e) =$ '(' and $endInter(q_c) =$ ')') then
   - // two queries have no intersection but q_c is consecutive to q_e
     - $q_c \leftarrow q_c \oplus q_e$
   - else
     - $\hat{l}_{temp} \leftarrow \hat{l}_{temp} \cup \{q_e\}$
   - end for
4. $\hat{l} \leftarrow \hat{l}_{temp} \cup \{q_c\}$

Figure 7.5: Query consolidation procedure.

larly, $endInter(q_a) = endInter(q_b)$ *iff* $(endInter(q_a) =$ ')' $\wedge$ $endInter(q_b) =$ ')') $\vee$ $(endInter(q_a) =$ ']' $\wedge$ $endInter(q_a) =$ ']').

Definition 3 *For two queries* q_a *and* q_b, $beginInter(q_a) \leq beginInter(q_b)$ *iff* $beginInter(q_a) < beginInter(q_b) \vee beginInter(q_a) = beginInter(q_b)$, *while* $endInter(q_a) \leq endInter(q_b)$ *iff* $endInter(q_a) < endInter(q_b) \vee endInter(q_a) = endInter(q_b)$. *In addition,* $beginInter(q_a) \not< beginInter(q_b)$ *iff* $\neg(beginInter(q_a) < beginInter(q_b))$, *while* $endInter(q_a) \not< endInter(q_b)$ *iff* $\neg(endInter(q_a) < endInter(q_b))$.

The detail for a member query q_m being consolidated with a query element q_e, i.e., $q_m \oplus q_e$, is as follows:

Scenario 1: there is no overlapping between the member query and the query element, which means that there is no intersection between two queries on the selection

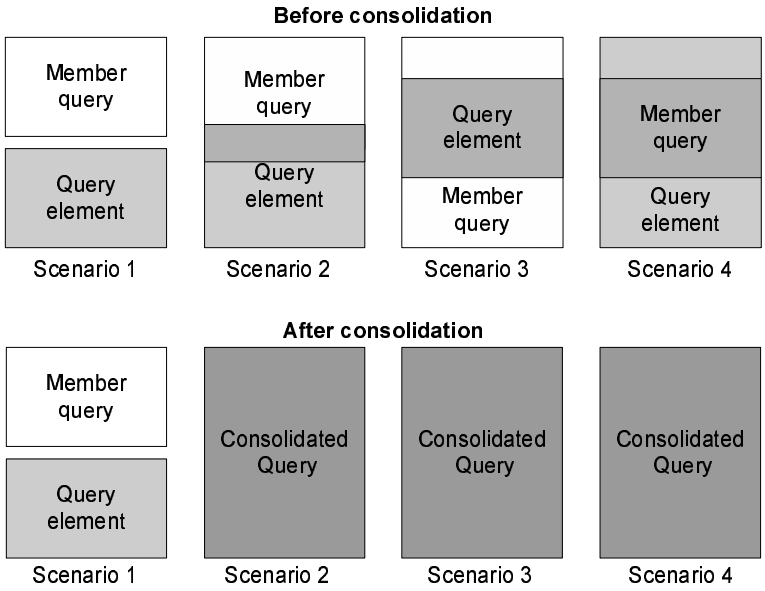


Figure 7.6: Possible scenarios in consolidating two queries.

attribute. There is no overlapping between two queries when $max(q_m) < min(q_e)$, or $min(q_m) > max(q_e)$, where $min(\cdot)$ and $max(\cdot)$ are functions for returning the minimum bound of a query and the maximum bound of a query respectively. In addition, two queries do not overlap even when $min(q_e) = max(q_m)$, if $beginInter(q_e)$ = '(' and $endInter(q_m)$ = ')'. Similarly, there is no intersection even when $min(q_m) = max(q_e)$, if $beginInter(q_m)$ = '(' and $endInter(q_e)$ = ')'. In all these cases, no single consolidated query can exactly covers both q_m and q_e. We will say that the result of the $q_m \oplus q_e$ operation is *undefined*.

On the other hand, two queries may have no intersection but are consecutive. This occurs when $min(q_a) = max(q_b)$, but $beginInter(q_a)$ = '(' and $endInter(q_b)$ = ']', or $beginInter(q_a)$ = '[' and $endInter(q_b)$ = ')', where q_a and q_b can be instantiated to q_m and q_e respectively, or q_e and q_m respectively. Two consecutive queries will be consolidated with the result range from the minimum of q_b to the maximum of q_a, i.e., $q_a \oplus q_b = \langle relation, attribute, \{beginInter(q_b), min(q_b), max(q_a), endInter(q_a)\}\rangle$. For discrete range queries the notion of consecutive query range is quite intuitive, and this can be modeled in the same way as above, by transforming, for example, an inclusive maximum bound into an exclusive maximum bound by the increment of maximum bound value by 1 before query consolidation. □

Scenario 2: the member query and the query element are partially overlapped. There are two situations involved in this scenario. In the first situation, a query segment of the member query on its "minimum" side does not overlap with the query element; the result of $q_m \oplus q_e$ will be $\langle relation, attribute, \{beginInter(q_m), min(q_m), max(q_e), endInter(q_e)\}\rangle$. There are a number of cases belonging to the first situation, as follows:

- $min(q_m) < min(q_e)$ and $min(q_e) < max(q_m) < max(q_e)$,
- $min(q_m) < min(q_e)$ and $min(q_e) < max(q_m) = max(q_e)$ and $endIter(q_m) < endIter(q_e)$,
- $min(q_m) < min(q_e)$ and $max(q_m) = min(q_e)$ and $endInter(q_m) =$ ']' and $beginInter(q_e) =$ '[',
- $min(q_m) = min(q_e)$ and $min(q_e) < max(q_m) < max(q_e)$ and $beginInter(q_m) < beginInter(q_e)$,
- $min(q_m) = min(q_e)$ and $max(q_m) = max(q_e)$ and $beginInter(q_m) < beginInter(q_e)$ and $endInter(q_m) < endInter(q_e)$

In the second situation, a query segment of the member query on its "maximum" side does not overlap with the query element; the result of $q_m \oplus q_e$ will be $\langle relation, attribute, \{beginInter(q_e), min(q_e), max(q_m), endInter(q_m)\}\rangle$. The cases that belonging to the second situation are as follows:

- $max(q_m) > max(q_e)$ and $min(q_e) < min(q_m) < max(q_e)$,
- $max(q_m) > max(q_e)$ and $min(q_e) = min(q_m) < max(q_e)$ and $beginInter(q_e) < beginInter(q_m)$,
- $max(q_m) > max(q_e)$ and $min(q_m) = max(q_e)$ and $endInter(q_e) =$ ']' and $beginInter(q_m) =$ '[',
- $max(q_m) = max(q_e)$ and $min(q_e) < min(q_m) < max(q_e)$ and $endInter(q_e) < endInter(q_m)$,
- $max(q_m) = max(q_e)$ and $min(q_m) = min(q_e)$ and $beginInter(q_e) < beginInter(q_m)$ and $endInter(q_e) < endInter(q_m)$

□

Scenario 3: the data range of the incoming member query totally covers the existing query element. In this scenario, the result member query will be the incoming member query, i.e., $q_m \oplus q_e = q_m$. Note that, in the last two cases, they are meaningful only when both equalities in q_e are inclusive. The cases belonging to this scenario are as follows:

- $min(q_m) < min(q_e)$ and $max(q_m) > max(q_e)$,
- $min(q_m) < min(q_e)$ and $min(q_e) < max(q_m) = max(q_e)$ and $endIter(q_m) \not< endIter(q_e)$,
- $max(q_m) > max(q_e)$ and $min(q_e) = min(q_m) < max(q_e)$ and $beginInter(q_e) \not< beginInter(q_m)$,
- $min(q_m) = min(q_e)$ and $max(q_m) = max(q_e)$ and $beginInter(q_m) \leq beginInter(q_e)$ and $endInter(q_e) \leq endInter(q_m)$,
- $min(q_m) = min(q_e) = max(q_e)$ and $beginInter(q_m) =$'[',
- $max(q_m) = max(q_e) = min(q_e)$ and $endInter(q_m) =$ ']'

□

Scenario 4: the data range of the existing member query is entirely covered by the query element. In this scenario, the result member query will be the query element, i.e., $q_m \oplus q_e = q_e$. In the last two cases, they are meaningful only when both equalities in q_m are inclusive. The cases belonging to this scenario are as follows:

- $min(q_e) < min(q_m)$ and $max(q_e) > max(q_m)$,
- $min(q_e) < min(q_m)$ and $min(q_m) < max(q_e) = max(q_m)$ and $endIter(q_e) \not< endIter(q_m)$,
- $max(q_e) > max(q_m)$ and $min(q_m) = min(q_e) < max(q_m)$ and $beginInter(q_m) \not< beginInter(q_e)$,
- $min(q_e) = min(q_m)$ and $max(q_e) = max(q_m)$ and $beginInter(q_e) \leq beginInter(q_m)$ and $endInter(q_m) \leq endInter(q_e)$,
- $min(q_e) = min(q_m) = max(q_m)$ and $beginInter(q_e) =$'[',
- $max(q_e) = max(q_m) = min(q_m)$ and $endInter(q_e) =$ ']'

□

Note that when the member query and the query element are exactly covered by each other, i.e., two queries have the same min, max, $beginInter$ and $endInter$ values, that fall within both Scenario 3 and 4. All these cases could be subsumed into either one of scenarios.

In the execution of the query consolidation procedure for an incoming member query q_m to the consolidated query list $\hat{l}_c$ (i.e., $consolidate(q_m, \hat{l}_c)$), as depicted in Figure 7.5, there is a temporary query list, $\hat{l}_{temp}$, for storing partial results in the execution of query consolidation procedure. The temporarily consolidated query q_c is initially assigned as the member query q_m. The query q_c is then consolidated with each query element q_e contained in the consolidated query list, $\hat{l}_c$. If there is no overlapping between the query element q_e and the partially consolidated query q_c and both queries are not consecutive with each other, the query element will be put into list $\hat{l}_{temp}$. Otherwise, both query q_c and query element q_e will be consolidated into a single query by the operation $q_c \oplus q_e$. The query consolidation procedure will continue to consolidate the partially consolidated query q_c with other query elements in $\hat{l}_c$. After all elements in the consolidated query list, $\hat{l}_c$, are examined, the final consolidated query q_c will be added into the list $\hat{l}_{temp}$ and returned as a new consolidated query list. Please note that each query list is sorted in ascending order according to the max value in the query range of each query element in the list. The purpose is for the correct execution for the query consolidation procedure and the supplementary query list generation discussed in the later sections. In addition, execution efficiency of those procedures could be improved because the procedures could be terminated earlier by eliminating the need of examining unrelated query elements.

Example #1: Considering a single relation database that contains the hotel information and an existing consolidated query list which contains elements {⟨Hotel, price, {[, 400, 600,]}⟩}, a new member query is received with the expression ⟨Hotel, price, {[,200, 500,]}⟩. Since there is only one element in the consolidated query list, the

new member query is to be consolidated with the element. According to the query consolidation procedure, the member query just received will be consolidated with the element by checking for its overlap with the query element. As the relationship between the member query and query element fits in Scenario 2, the result query list after consolidation will be ⟨Hotel, price, {[, 200, 600,]}⟩. ■

Example #2: Suppose that the existing consolidated query list contains elements {⟨Hotel, price, {[,300, 400,]}⟩, ⟨Hotel, price, {[,500, 600,]}⟩, ⟨Hotel, price, {[,900, 1000,]}⟩}. The group leader receives a member query q_m = ⟨Hotel, price, {[,200, 500,]}⟩. By performing query consolidation procedure, q_m is consolidated with the elements in the existing consolidated query list. The result query list after query consolidation will be {⟨Hotel, price, {[,200, 600,]}⟩, ⟨Hotel, price, {[,900, 1000,]}⟩}. The detail for executing the query consolidation procedure for the member query q_m in this example is depicted in Figure 7.7. ■

Example #3: Suppose that the existing consolidated query list contains elements {⟨Hotel, price, {[, 400, 600,]}⟩}, a new member query is received with the expression ⟨Hotel, price, {(,600, 700,]}⟩. Since there is only one element in the consolidated query list, the new member query just received will be consolidated with the element by checking for its overlap with the query element in the execution of the query consolidation procedure. It is found that two queries do not overlap but are consecutive. As the relationship between the member query and query element fits in Scenario 1, the result query list after consolidation will be ⟨Hotel, price, {[, 400, 700,]}⟩. ■

7.2.2 Query Listening Period Adjustment

After the new consolidated query list is generated, the query listening period will be adjusted, according to the coverage of the outstanding query list to the new consolidated query list. The basic idea of query listening period adjustment is to have a basic listening period for collecting and consolidating a reasonable amount of queries from

Existing consolidated query list:

$\hat{l}_c = \{\langle$Hotel, price, {[,300, 400,]}$\rangle$, $\langle$Hotel, price, {[,500, 600,]}$\rangle$,

$\langle$Hotel, price, {[,900, 1000,]}$\rangle\}$

Upon receiving member query, q_m: $\langle$Hotel, price, {[,200, 500,]}$\rangle$

Executing the query consolidation procedure:

Initialization: $q_c \leftarrow q_m$

$\hat{l}_{temp} = \emptyset$, $q_c = \langle$Hotel, price, {[,200, 500,]}$\rangle$

After 1st round: q_c consolidated with $q_{e1} = \langle$Hotel, price, {[,300, 400,]}$\rangle$

$\hat{l}_{temp} = \emptyset$, $q_c = \langle$Hotel, price, {[,200, 500,]}$\rangle$

After 2nd round: q_c consolidated with $q_{e2} = \langle$Hotel, price, {[,500, 600,]}$\rangle$

$\hat{l}_{temp} = \emptyset$, $q_c = \langle$Hotel, price, {[,200, 600,]}$\rangle$

After 3rd round: $q_c \cap q_e = \emptyset$, where $q_{e3} = \langle$Hotel, price, {[,900, 1000,]}$\rangle$

$\hat{l}_{temp} = \{\langle$Hotel, price, {[,900, 1000,]}$\rangle\}$,

$q_c = \langle$Hotel, price, {[,200, 600,]}$\rangle$

After the query consolidation procedure:

$\hat{l}_c = \{\langle$Hotel, price, {[,200, 600,]}$\rangle$, $\langle$Hotel, price, {[,900, 1000,]}$\rangle\}$

Figure 7.7: Execution of query consolidation procedure in Example #2.

group members. If it is found that a high proportion of the consolidated query list could be covered by the outstanding query list stored in the group leader, this significant portion of query could be answered in the next data broadcast. Thus, the leader could adjust to wait for a longer period for collecting more member queries in the next cycle.

Before discussing the query listening period adjustment, the notion of *coverage* will be introduced. The coverage is the proportion of consolidated query list covered by the outstanding query list. To obtain the coverage value for each query element, q_i, in a consolidated query list, the sub-lists in the query element that are overlapped with the query elements in the outstanding query list are determined. Then, the ranges of the sub-lists are added together and this range total is considered the coverage length, ζ_{q_i}, for this consolidated query element. After the coverage length for every consolidated

query list element is calculated, they will be summed together as the total coverage length ζ_{cTotal} for the consolidated query list. Meanwhile, the range total of the consolidated list, $\zeta_{\hat{l}_c}$, is obtained. The ratio $\frac{\zeta_{cTotal}}{\zeta_{\hat{l}_c}}$ will be the *coverage* that a consolidated query list being covered by an outstanding query list. The coverage value would range from 0 to 1. As the coverage value increases, it represents a larger portion of consolidated query list being covered by the outstanding query list. In other words, the portion that the supplementary query list is generated out of the consolidated query list would be smaller. The query listening period could thus be increased for consolidating more member queries. Since each query element is a range query segment, there is no overlapping between query elements in a query list. The summation of all the coverage lengths of all query elements in the consolidated query list should be smaller than the summation of the range of all query elements in the consolidated query list. To illustrate the calculation of coverage, consider Figure 7.8, in which an outstanding query list contains two query elements q_A and q_B and a consolidated query list contains two query elements q_1 and q_2. To calculate the coverage for the consolidated query list, the sub-lists of each query element in the consolidated query list are obtained and their corresponding coverage lengths are calculated. For the query element q_1, its coverage length is 70, i.e., $\zeta_{q_1} = (100 - 80) + (200 - 150) = 70$. For the query element q_2, its coverage length is 30, i.e., $\zeta_{q_2} = 260 - 230 = 30$. Thus, the total coverage length ζ_{cTotal} for the consolidated query list is 100. As the range total of the consolidated list is 160, where $\zeta_{\hat{l}_c} = (200 - 80) + (270 - 230) = 160$, the coverage is $\frac{\zeta_{cTotal}}{\zeta_{\hat{l}_c}} = \frac{100}{160} = 0.625$.

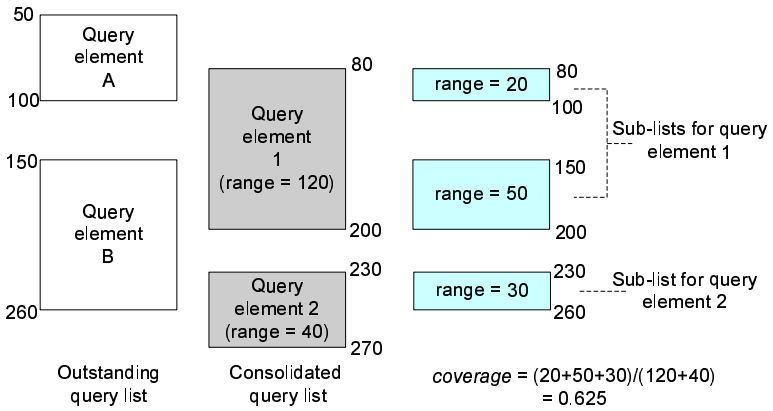


Figure 7.8: Coverage calculation.

Whenever a new member query is received and consolidated, the new coverage for the new consolidated query list will be calculated. After the new coverage is obtained, the query listening period is also adjusted accordingly. As a heuristic, the listening period should be increased when the coverage value becomes higher. Since the increase of coverage value means larger portion of query elements in the consolidation query list covered by the outstanding query list, this implies that this portion will be answered sooner or later from the server response. The query listening period can be adjusted to listen and consolidate more member queries, according to the equation: $P_{current} = P_{basic} \times (1 + coverage)$, where $P_{current}$ is the current query listening period value while P_{basic} is a basic listening period value. According to this arrangement, the value of $P_{current}$ will be bounded between P_{basic} to $2 \times P_{basic}$ for avoiding large increase of the query response time of a query by the high value of query listening period.

Example #4: Continuing with Example #2, the leader stores the consolidated query list: $\{\langle$Hotel, price, {[,300, 400,]}$\rangle$, $\langle$Hotel, price, {[,500, 600,]}$\rangle$, $\langle$Hotel, price, {[,900, 1000,]}$\rangle\}$ before receiving the member query $q_m = \langle$Hotel, price, {[,200, 500,]}$\rangle$. Suppose there is an outstanding query list containing a query element: $\{\langle$Hotel, price, {[,800, 950,]}$\rangle\}$. The range covered by the outstanding query list in the consolidated query list is between 900 and 950 and the total coverage length, ζ_{cTotal}, in the consolidated query list is 50. The summation of the ranges of all query elements is $\zeta_{\hat{l}_c} = (1000 - 900) + (400 - 300) + (600 - 500) = 300$. The coverage equals $50/300 = 1/6$. Assuming that the basic query listening period is 120s, the query listening period before consolidating with the new member query q_m will be $120s \times (1 + 1/6) = 140s$. After q_m is received and consolidated, the consolidated query list now contains elements $\{\langle$Hotel, price, {[,200, 600,]}$\rangle$, $\langle$Hotel, price, {[,900, 1000,]}$\rangle\}$. The range covered by the outstanding query list is unchanged while the summation of the ranges from all consolidated query list elements is changed as $\zeta_{\hat{l}_c} = (600 - 200) + (1000 - 900) = 500$. Note that the increase in the summation value leads to the decrease in the query listening period value. The new coverage

after consolidating with member query q_m is now $50/100 = 1/10$ and the new query listening period is adjusted to $120s \times (1 + 1/10) = 132s$. The query listening period adjustment for an incoming member query q_m is visualized in Figure 7.9. ■

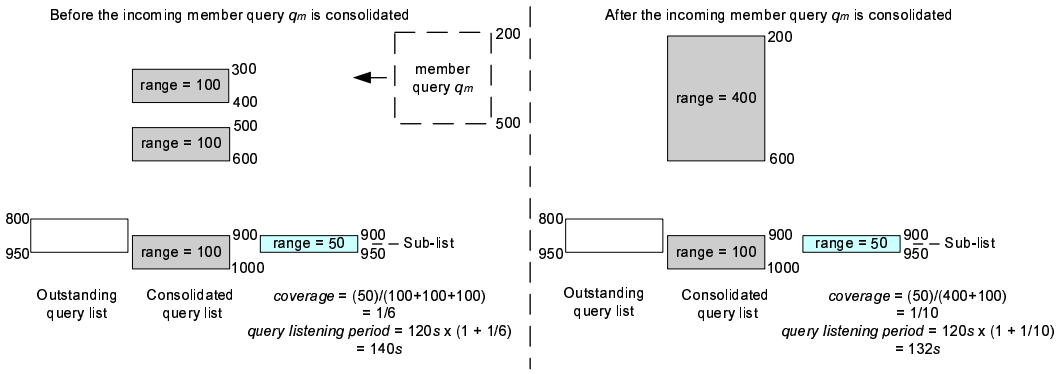


Figure 7.9: Query listening period adjustment in Example #4.

7.2.3 Global Query Request

Timing starts when a member query is inserted into an empty consolidated query list as the first element in the list. The time value will be recorded as $consolidation\ \ start\ \ time$, t_c. The leader will be notified when the query listening period expires, i.e., $now - t_c > P_{current}$. A supplementary query list will be compiled and sent to the server. To generate the supplementary query list from a consolidated query list, as depicted in Figure 7.10, each query element in the consolidated query list is compared with the query elements in the outstanding query list. The query segments from the query element that are not overlapped with any query element in the outstanding query list will be inserted into a sub-list. After all non-overlapping query segments for that consolidated query list element are discovered, the elements in the sub-list will be merged into the existing supplementary list. After the process is finished, the supplementary query list contains query elements from the consolidated query list not covered by the outstanding query list. However, if the supplementary query list generated is empty, no action would be performed, except that the timer is reset. An empty supplementary query list means that the queries stored in the consolidated list could be completely answered from the next data broadcast. After the supplementary query list is sent to

the server from a group leader, the existing outstanding list will be updated into a new one, by merging the existing outstanding query list with the consolidated query list. In addition, the supplementary query list will be cleared after the list is sent. Recalling Example #4, the supplementary query list to be sent to the server will be {⟨Hotel, price, {[,200, 600,]}⟩, ⟨Hotel, price, {(,950,1000,]}⟩}, while the new outstanding list will be {⟨Hotel, price, {[,200, 600,]}⟩, ⟨Hotel, price, {[,800, 1000,]}⟩}.

Supplementary query list generation:

1. Clear the supplementary list, $\hat{l}_s \leftarrow \emptyset$.
2. for each query element q_i in $\hat{l}_c$ do
   compare q_i with each element q_j in outstanding query list $\hat{l}_u$
   extract the query segments from q_i that are not overlapped with the elements in $\hat{l}_u$
   put those query segments into a sub-list, $\hat{l}_{sub}$
   $\hat{l}_s \leftarrow merge(\hat{l}_{sub}, \hat{l}_s)$
   clear the sub-list, $\hat{l}_{sub} \leftarrow \emptyset$
   end for

Figure 7.10: Supplementary query list generation.

After a group leader sends the supplementary query list to the server and a new outstanding list is generated, the leader listens to the index of data broadcast for data items to be broadcast that match with the query elements in the outstanding query list. The outstanding list will be adjusted according to the index information received. The query element portion that could be answered from the data broadcast will be pruned from the outstanding query list. In addition, the members that have submitted member queries to the leader will also listen for the arrival of their interested data items from data broadcast according to the index disseminated.

7.3 Performance Study

Simulation study is conducted for investigating the performance effect of group-based data management. All mobile hosts move freely according to the random waypoint model. Summary of the system characteristics for the group-based model can be referred in Table 6.1. Mobile host are clustered into a set of groups based on our group-

Table 7.1: System characteristics in the experiments.

Parameter	Default value
Number of tuples:	65536
Tuple size:	208 bytes
Selectivity, σ:	5%
Query inter-arrvial time, $1/\mu$:	$5s$
Basic query listening period:	$20s$
Broadcast channel bandwidth, γ:	2Mbps
Long-range uplink data transfer rate, $\hat{T}_{rate\_long}$:	153.6Kbps
Short-range uplink data transfer rate, $\hat{T}_{rate\_short}$:	153.6Kps – 5Mbps

based model and the group-based framework (see Chapters 3 and 4). A simple scenario is considered in the simulation. In the simulated environment, there is a database with one relation containing 65536 tuples. Each tuple has a size of 208 bytes [13]. Each mobile host generates a range query after a random period, according to the Possion distribution at a rate of μ, i.e., an query inter-arrival time of $1/\mu$. It is $5s$ in the default setting. For simplicity, mobile hosts will only select a single same attribute in the simulation. Each range query generated will be bounded by a minimum value that is picked up from 65536 tuples randomly, while the maximum value of the range query is determined by the selectivity parameter, σ, defined in the simulation with a default value 5%. Selectivity is the percentage of tuples to be queried out of the total number of tuples in the database. The server modeled in the simulation receives queries from the mobile hosts. After a query is received, the server will retrieve the data for the query and broadcast the data to the mobile hosts. The length of a data broadcast will be determined by the data size to be transmitted in answering the query received and the bandwidth of the broadcast channel, γ, which is 2Mbps in the simulation. The experimental setting is summarized in Table 7.1. We compare the performance of our GBD approach with individual-based data management, in which each mobile host generates queries and directly sends them to the server, in terms of number of messages to server, total message size to be transmitted to the server (in terms of total number of bytes transmitted to server) and the total communication cost for different schemes.

7.3.1 Performance of GBD with Different Basic Query Listening Periods

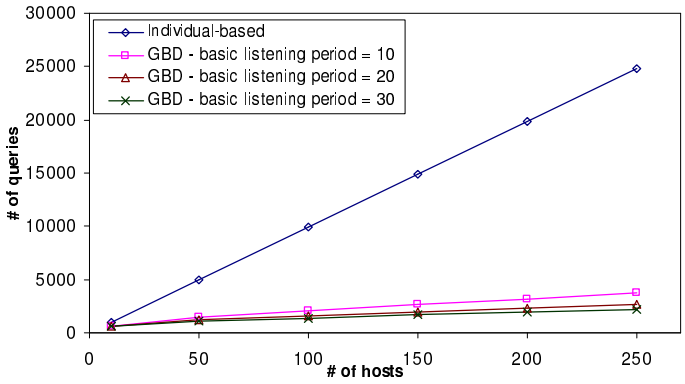


Figure 7.11: The performance effect of GBD in the number of queries to server.

The first set of experiments is to study the effectiveness of the GBD scheme with different basic query listening periods. With short query listening period, supplementary query lists will be sent to the server more quickly, thus latency is reduced. In the first experiment, we study the performance of GBD in terms of total number of uplink queries being sent to server. From Figure 7.11, it is obvious that our approach drastically reduces the number of messages to the server for all basic listening periods. There is smaller differences between individual-based approach and our GBD approach in low population environments. It is because it is difficult for a mobile host to join a group and many singleton groups are formed. This leads to many queries being directly sent to the server without performing any consolidation. As the population density increases, more non-singleton groups are formed and the benefit of the GBD scheme becomes more significant. As expected, as the basic query listening period decreases, the number of messages to server increases but there is only a small increment in the query message count relative to the number of queries in the individual-based approach. In short, the GBD scheme is effective in reducing the number of query messages sent to the server.

We also study the total message size, in terms of number of bytes, being sent to the server with different approaches. As depicted in Figure 7.12, it is obvious that the total message size in individual-based approach is larger than that in the GBD approach.

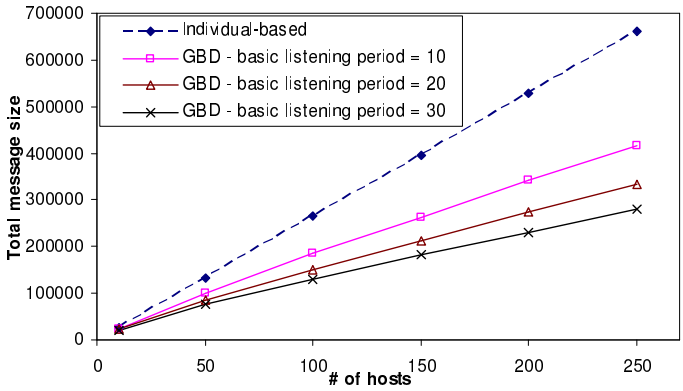


Figure 7.12: The performance effect of GBD in the total message size.

Similar to the number of queries in low population environments, the difference in total message size between the individual-based approach and the GBD approach is smaller. There are significant decreases in the total message size in high population environments. Consistent with the number of uplink messages to server, the total message size in GBD approach increases when the basic listening period decreases across different host populations. As the basic listening period decreases, group leaders have less time to collect and consolidate member queries, i.e., a leader receives fewer member queries. In turn, the probability of overlapping among the member queries decreases, resulting in a smaller savings via consolidation.

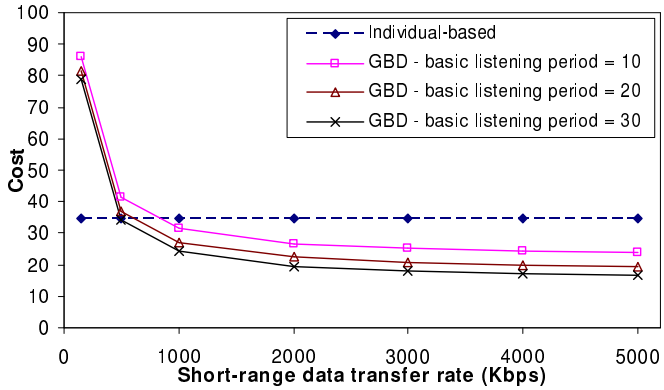


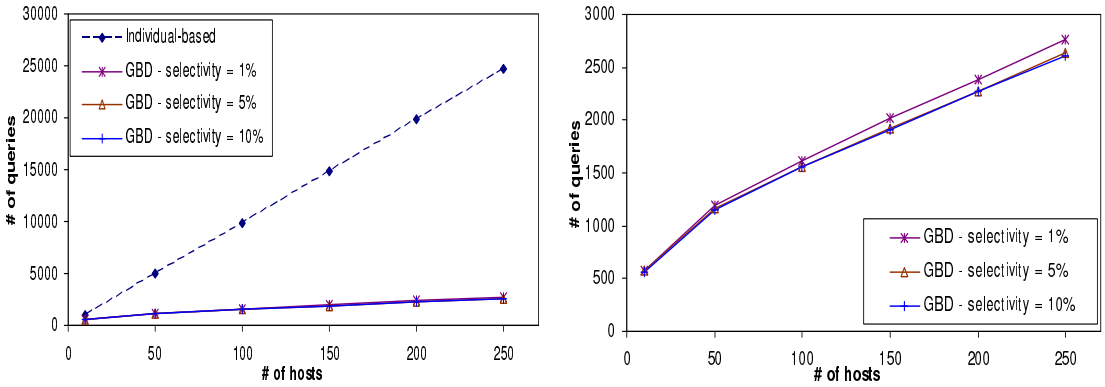
Figure 7.13: Communication cost of the GBD scheme in different basic periods.

To study the performance of the GBD from the collective view of all mobile hosts, both short-range and long-range communication costs are be taken into account. Here, the cost function for sending a long-range message is modeled as $\hat{C}_{startup\_long} + \frac{\hat{D}^i_{long}}{\hat{T}_{rate\_long}}$, where $\hat{C}_{startup\_long}$ is the startup cost for sending a long-range message, $\hat{D}^i_{long}$ is the message size of a long-range message i and $\hat{T}_{rate\_long}$ is the data transfer rate of

the long-range uplink channel. Similarly, the cost for sending a short-range message is $\hat{C}_{startup\_short} + \frac{\hat{D}^j_{short}}{\hat{T}_{rate\_short}}$, where $\hat{C}_{startup\_short}$ is the startup cost for sending a short-range message, $\hat{D}^j_{short}$ is the message size of a short-range message j and $\hat{T}_{rate\_short}$ is the data transfer rate of the short-range communication channel. In the simulation, it is assumed that both $\hat{C}_{startup\_long}$ and $\hat{C}_{startup\_short}$ can be neglected as we would like to focus on the effect of message size to the GBD scheme. In the GBD scheme, a portion of local messages is for the purpose of realizing the group-based model, which includes group management, leadership maintenance and intra-group location management. This kind of messages are known as *core message*. It is assumed that a constant message size, S_G = 15 bytes, is required for the passing a core message. The total aggregated cost will be $\hat{C}_{GBD} = \frac{\sum_{i=1}^{|Q_l|} \hat{D}^i_{long}}{\hat{T}_{rate\_long}} + \frac{|\hat{M}_G| \times S_G + \sum_{j=1}^{|Q_s|} \hat{D}^j_{short}}{\hat{T}_{rate\_short}}$, where $|\hat{M}_G|$ and $|Q_s|$ represent the total number of core messages and the total number of member queries issued to leaders respectively, while $|Q_l|$ is the total number of long-range uplink messages being sent to the server. The performance effect in terms of communication cost is studied with a mobile population of 250 with long-range uplink data transfer rate $\hat{T}_{rate\_long}$ = 153.6Kbps [42] and different short-range data transfer rates $\hat{T}_{rate\_short}$, from 153.6Kbps to 5Mbps, are investigated.

From Figure 7.13, it is obvious that the cost for the individual-based scheme remains constant, since it involves no short-range message cost. However, the cost of the GBD schemes in different basic listening period decreases when $\hat{T}_{rate\_short}$ value increases. The GBD scheme starts to outperform the individual-based approach at low-medium short-range data transfer rates. In general, the GBD scheme with a high basic listening period yields a lower aggregated cost than that with a low basic listening period. This is consistent with our previous experiments that the GBD scheme achieves a better performance with a longer basic listening period. It is because more member queries can be collected with a longer basic listening period, thus increasing the chance that more member queries can be consolidated, in turn reducing the number of discrete query elements in the supplementary query list.

7.3.2 Performance of GBD with Different Selectivities



(a) (b) Result on the left at a larger scale

Figure 7.14: The performance effect of GBD in the number of queries to server.

To investigate the effect of query selectivity with the GBD scheme, another set of experiments is conducted, with different value settings in selectivity. The number of uplink queries is also studied, as depicted in Figure 7.14. Our GBD scheme outperforms the individual-based approach with different selectivities, as depicted in Figure 7.14(a). From Figure 7.14(b), there is a slightly increase in the number of uplink queries with low selectivity (σ = 1%). It is because the chance to have non-overlapping query elements between a consolidated query list and an outstanding query list increases at low selectivity. In turn, the coverage of the consolidated query list to the outstanding list decreases. In terms of total message size, as depicted in Figure 7.15, the total message size increases as the selectivity decreases. The fact is that more non-overlapping query elements are generated at low selectivity because of the decrease in the coverage between the consolidated query list and the outstanding query list.

The aggregated cost performance on the GBD scheme with different selectivities is studied in a population of 250 mobile hosts. We adopt the total cost functions for the individual-based approach and the GBD scheme defined in Section 7.3.1. The GBD scheme begins to outperform the individual-based approach in low-medium short-range data transfer rates, as depicted in Figure 7.16. As expected, the cost in the GBD scheme increases with decrease in query selectivity. It is because more discrete query elements are contained in the supplementary query list in low selectivity. This results

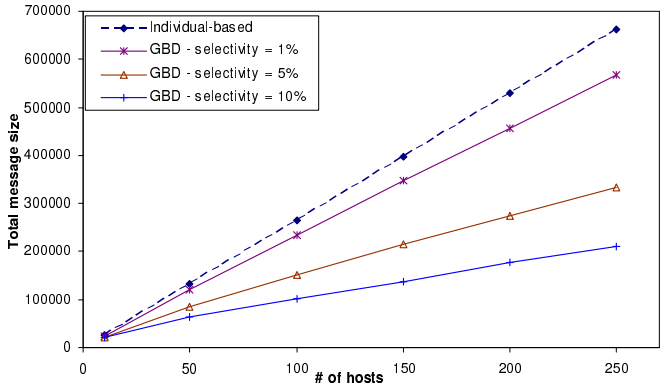


Figure 7.15: The performance effect of GBD in the total message size.

in larger message size in sending all query elements in a supplementary query list to the server in a single message. This fact is reflected in the GBD scheme with selectivity $\sigma = 1\%$, in which the scheme starts to yield a better result than individual-based scheme at medium to high data transfer rate.

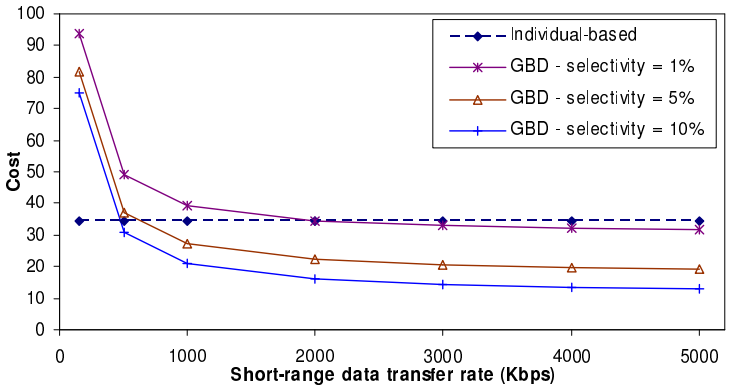
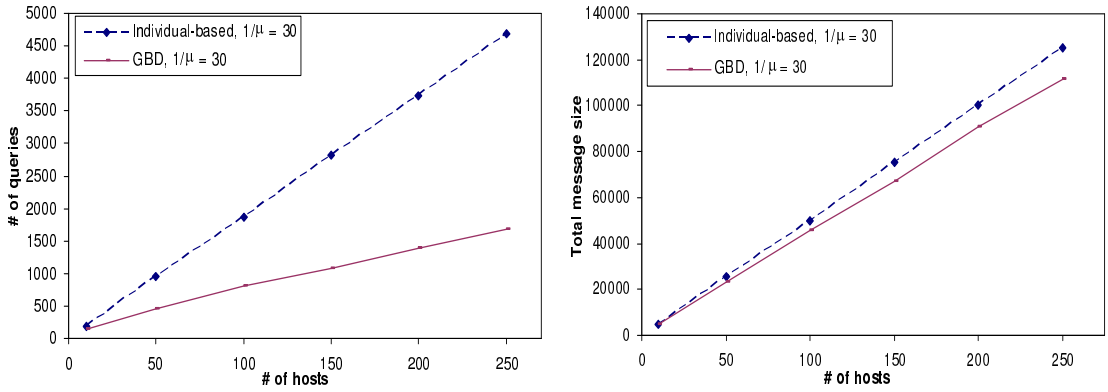


Figure 7.16: Communication cost of the GBD scheme in different selectivities.

7.3.3 Performance of GBD in Low Query Frequency Environment

We also evaluate the effect in the GBD scheme in low query frequency environment. In this set of experiments, the query inter-arrival time, $1/\mu$, is set to $30s$. As expected, the number of queries to the server decreases in both individual-based approach and GBD scheme with the decrease in query frequency, as depicted in Figure 7.17(a). Consistent with results in previous set of experiments, the GBD scheme outperforms the individual-based approach in terms of number of uplink queries almost in all population environments, especially in high population environments. In terms of total



(a) # of queries (b) Total message size

Figure 7.17: The performance effect of GBD in # of queries and total message size.

message size transmitted to server, as depicted in Figure 7.17(b), the decrement in the total message size in the GBD scheme is relatively small when compared with high query frequency environment. It is expected because the chance for a member query that can be overlapping with the query elements in the consolidated list decreases in low query frequency. The same situation happens between the consolidated query list and the outstanding query list. As fewer member queries can be received for a group leader, the probability that the consolidated query list contains fewer query elements for overlapping with the query elements in the outstanding query list will be lower; a supplementary query list with more query elements will be generated, in turn increasing the message size of a query to server in the GBD scheme. However, the GBD scheme is still effective in reducing the total message size transmitted to the server.

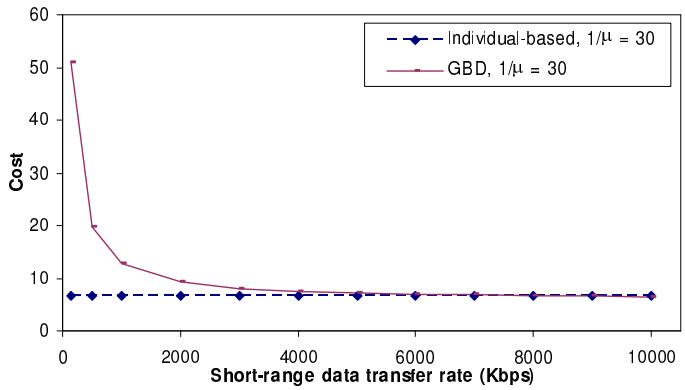


Figure 7.18: Communication cost of the GBD scheme in different selectivity.

For studying the aggregated cost performance in low query frequency, the total cost functions for the individual-based approach and the GBD scheme defined in Sec-

tion 7.3.1 are adopted. A population with 250 mobile hosts was simulated. As depicted in Figure 7.18, the GBD scheme could only yield a slight performance improvement to the individual approach at high short-range data transfer rate. This is caused by several factors. With low query frequency, the local communication cost in realizing the group-based model becomes significant. In realizing the group-based model, collaboration between members and the leader in a group is required for group management, leadership maintenance and local level of location management. The amount of core messages contributes a certain proportion of local communication. Another factor is that there is only a smaller decrement in the total message size in the GBD scheme, as depicted in Figure 7.17(b). Larger total uplink message size is generated when compared with that in high query frequency environment. To enhance the performance in the GBD scheme at low query frequency, piggybacking technique can be employed, as mentioned in Section 3.2. Queries issued by members could be piggybacked in the core messages, and vice versa. Performance can also be improved when an adaptive listening period adjustment mechanism is employed, which can defer the timeout of the listening period after sufficient discrete elements in the supplementary list are discovered, so that more member queries can be consolidated. Finally, a thorough integration of the GBD scheme to the proposed framework will be considered in near future.

Chapter 8

Conclusion

Group-based paradigm provides a novel way for tackling the issues in location and data management in mobile environments, especially in reducing the volume of expensive uplink traffic. In this thesis, a group-based model is formulated and a group-based framework is proposed for facilitating the development of mobile applications according to the group-based paradigm. Key components such as group formation, group management, leadership maintenance and location management component, form the foundation of the group-based framework.

The group-based location management (GBL) scheme, which is built on top of the group-based framework, increases scalability because the workload of the moving object database server is decreased through a reduction on the volume of uplink traffic. The tradeoff is that the load on a mobile host may increase because of its need to participate in group membership maintenance, by handling local communication. We studied location updating strategies for location updating from group members to their leaders and for location updating from leaders to the stationary location server. Simulation results show that GBL outperforms conventional individual-based schemes, such as *plain dead-reckoning*, in terms of reducing the number of costly update uplink messages sent in general mobile environments.

Group management is one of the core components for realizing the group-based

framework. Membership maintenance is important to the performance of the system, and improvement works on this have been conducted. Efficient group finding process is also critical in the performance of group management, which could contribute to a significant portion of local communication overhead, especially in a high host population environment, with higher number of groups and size of a group. Thus, in this thesis, we proposed two improvements on the group finding process, so as to reduce the number of local messages generated. As a side effect, the load on non-leader mobile hosts within a group is greatly reduced. Results from our simulation study in the GBL context indicate that our leader-filter join procedure outperforms the basic approach in reducing local communication overhead effectively, though there is a slight increase in the number of group location update messages with a medium filtering threshold value. However, the negative impact in group location updating is comparatively small since GBL with a medium filtering threshold in an improved join procedure still outperforms the individual-based approach. From the mobile hosts' point of view, the proposed leader-filter join procedure is effective in alleviating their workload. To further reduce the local communication overhead, piggybacking technique could be introduced. The local messages for group management could be piggybacked in the messages issued by the applications, and vice versa. In addition, the time period to trigger group finding process for a singleton group can adapt according to the knowledge from the environments. For example, if a singleton group member could not find a suitable group to join after a certain number of group finding processes, it could extend the time period for the next group finding process.

Another integral part in the framework is leadership maintenance. Maintaining the leadership of a group properly is an important tactic in preserving the group stability and in enhancing the performance of group-based location updating scheme. We proposed a leadership maintenance scheme by employing the notion of stand-by secondary leader. The secondary leader is dynamically selected in the course of system execution, and it will be able to take over the job of its primary counterpart as soon as possible when the primary departs from the group, by executing the *turnover pro-*

cedure. The turnover procedure can also be invoked when a tend-to-leave leader is identified by the *turnover activation policy*. Since there is always a secondary leader standing by, the duration of leader absence is basically eliminated. Simulation study of GBL scheme with different leadership maintenance approaches indicates that our leadership maintenance approach outperforms a straightforward approach under GBL, which re-executes the mobile ad hoc network clustering algorithm upon a leader departure event, in reducing the number of group location update messages, the average number of groups and the aggregated cost. An improved variant of our scheme can further reduce the aggregated cost.

Group-based data management (GBD) provides a novel approach for querying data from mobile hosts to stationary information server based on the group-based model proposed. Member queries are collected and consolidated at group leaders. Query consolidation procedure is proposed for a group leader to consolidate received member queries so that overlapping queries could be eliminated. A query listening period adjustment policy is introduced for adjusting the time period for collecting member queries according to the coverage of the outstanding query list with respect to the consolidated query list. After the query listening period expired, a final supplementary query list will be sent to the information server for the interested data items. Simulation is conducted for studying the performance effect of the GBD scheme with traditional individual-based data management approach. Simulation results show that our approach outperforms the individual-based one in terms of number of queries, and total message size being sent to the server. The GBD scheme also yields a better aggregated cost performance in high query frequency environment.

When comparing the individual-based paradigm with the group-based paradigm in location and data management, the group leader seems to be a heavily burdened entity among the members in their groups, because it needs to perform additional task in sending uplink messages. However, we notice that those group leaders also need to send expensive uplink messages to the server in the individual-based approach anyway. In addition, mostly a leader acts as a receiver rather than a sender in the group-based

paradigm, and it is less costly for a receiver receiving a message than a sender sending a message. Thus, the load of a group leader in the group-based paradigm should not be too much heavier than the load in the individual-based approach. For improvement, the energy factor can be included in secondary leader determination algorithm to favor those with reasonable amount of energy remaining. The energy factor can also be integrated with existing turnover activation policy. A weight-based function could be introduced to take into account of both the $stayIndex$ and the energy remaining. The leader would turn over its job to the secondary leader if the result of the weight-based function is lower than a threshold value.

To further extend our work, a prototype for the group-based framework could be implemented to investigate precisely the effectiveness of the group-based framework in real world application. It is believed that the effectiveness of the group-based framework becomes significant when more mobile applications are built on top of it. Mobile applications suitable for the group-based framework will be developed. An interesting application of the framework is to provide a data caching mechanism among members within a group. Further expensive uplink messages could be reduced when the GBD scheme is supplemented with the mechanism. Performance effect of the framework when more group-based applications are executing simultaneously will be studied.

To enhance the effectiveness of the GBD scheme, a thorough integration of the GBD scheme with the group-based framework could be done. Piggybacking technique could be adopted for further reduction in both short-range and long-range communication overheads. A dynamic policy on query listening period adjustment can be extended for increasing the adaptivity of the GBD scheme.

Further study will be conducted on the algorithms and the schemes composing the core module in the framework on environments where the assumption on the rare disconnection in the communication between mobile hosts is relaxed. Investigation on the performance effect to different movement models, such as group mobility movement model, will be performed. Experiments are to be conducted for studying the perfor-

mance of the GBD scheme in hot spot query environments. On the other hand, the improvement of the consolidated procedure in the GBD scheme for supporting more query operations and multi-attribute selection queries will be investigated, alongside semantic query processing.

Bibliography

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast disks: Data management for asymmetric communication environments. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 199–210. ACM Press, 1995.

[2] A. Agarwal and S. R. Das. Dead reckoning in mobile ad hoc networks. In *Proceedings of the 2003 IEEE Wireless Communications and Networking Conference*, pages 1838–1843, Mar 2003.

[3] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 175–186. ACM, 2000.

[4] B. An and S. Papavassiliou. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks. *International Journal of Network Management*, 11(6):387–395, 2001.

[5] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the 19th IEEE INFOCOM, vol. 2*, pages 775–784, 2000.

[6] D. Barbara and T. Imielinski. Sleepers and workaholics: Caching strategies in mobile environments (extended version). *The VLDB Journal*, 4(4):567–602, 1995.

[7] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, Inc., 1996.

[8] E. Cayirci and I. F. Akyildiz. User mobility pattern scheme for location update and paging in wireless systems. *IEEE Transactions on Mobile Computing*, 1(3):236–247, Jul-Sep 2002.

[9] B. Y. L. Chan, A. Si, and H. V. Leong. Cache management for mobile databases: Design and evaluation. In *Proceedings of the Fourteenth International Conference on Data Engineering*, pages 54–63. IEEE Computer Society, 1998.

[10] W. W. Chang and H. J. Schek. A signature access method for the starburst database system. In *Proceedings of the Fifteenth International Conference on Very Large Data Bases*, pages 145–153. Morgan Kaufmann Publishers Inc., 1989.

[11] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, 5(2):193–204, 2002.

[12] M. Choy, M. Kwan, and H. V. Leong. Distributed database design for mobile geographical applications. *Journal of Database Management*, 11(1):3–15, Jan 2000.

[13] D. J. DeWitt. The wisconsin benchmark: Past, present, and future. In J. Gray, editor, *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*, pages 269–315. Morgan Kaufmann, 1993.

[14] C. Faloutsos and S. Christodoulakis. Signature files: An access method for documents and its analytical performance evaluation. *ACM Transactions on Information Systems*, 2(4):267–288, 1984.

[15] E. Gafni and D. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1):11–15, Jan 1981.

[16] R. H. Güting. An introduction to spatial database systems. *The Very Large Data Bases Journal*, 3(4):357–399, Oct 1994.

[17] J. Haartsen, M. Naghshineh, J. Inouye, O. J. Joeresson, and W. Allen. Bluetooth: Vision, goals, and architecture. *ACM Mobile Computing and Communications Review*, 2(4):38–45, Oct 1998.

[18] Z. J. Haas and B. Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, Apr 1999.

[19] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2/3):187–197, 2002.

[20] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, Aug 2001.

[21] X. Hong and M. Gerla. Dynamic group discovery and routing in ad hoc networks. In *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop*, pages 53–60, Sep 2002.

[22] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless network. In *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 53–60, Aug 1999.

[23] H.-Y. Hsieh and R. Sivakumar. On using the ad-hoc network model in cellular packet data networks. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 36–47. ACM Press, 2002.

[24] H.-Y. Hsieh and R. Sivakumar. Towards a hybrid network model for wireless packet data networks. In *Proceedings of the Seventh International Symposium on*

Computers and Communications (ISCC'02), pages 264 – 271. IEEE Computer Society, 2002.

[25] J.-L. Huang, M.-S. Chen, and W.-C. Peng. Exploring group mobility for replica data allocation in a mobile environment. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 161–168. ACM Press, 2003.

[26] Y. Huh and C. Kim. Group-based location management scheme in personal communication networks. In *Proceedings of the 16th International Conference on Information Networking*, pages 81–90, Jan 2002.

[27] T. Imielinski and B. R. Badrinath. Querying in highly mobile distributed environments. In *Proceedings of the 18th Conference on Very Large Databases*, pages 41–52, 1992.

[28] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on air: Organization and access. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):353–372, 1997.

[29] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[30] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 66–75. ACM Press, 1998.

[31] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4):422–469, 2000.

[32] K.-Y. Lam, O. Ulusoy, T. S. H. Lee, E. Chan, and G. Li. An efficient method for generating location updates for processing of location-dependent continuous queries. In *DASFAA '01: Proceedings of the 7th International Conference on*

Database Systems for Advanced Applications, pages 218–225. IEEE Computer Society, 2001.

[33] C.-H. Lee and M.-S. Chen. Processing distributed mobile queries with interleaved remote mobile joins. *IEEE Transactions on Computers*, 51(10):1182–1195, 2002.

[34] K. C. K. Lee, H. V. Leong, and A. Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[35] K. C. K. Lee, H. V. Leong, and A. Si. Semantic data access in an asymmetric mobile environment. In *Proceedings of the Third International Conference on Mobile Data Management*, pages 94–101. IEEE Computer Society, 2002.

[36] K. C. K. Lee, H. V. Leong, and A. Si. Semantic data broadcast for a mobile environment based on dynamic and adaptive chunking. *IEEE Transactions on Computers*, 51(10):1253–1268, 2002.

[37] K. C. K. Lee, H. V. Leong, and A. Si. Approximating object location for moving object database. In *Proceedings of 2003 International Workshop on Mobile Distributed Computing (ICDCS Workshop)*, pages 402–407, May 2003.

[38] W.-C. Lee and D. L. Lee. Signature caching techniques for information filtering in mobile enviroments. *Wireless Networks*, 5(1):57–67, 1999.

[39] G.-H. Li, K.-Y. Lam, T.-W. Kuo, and S.-W. Lo. Location management in cellular mobile computing systems with dynamic hierarchical location databases. *Journal of Systems and Software*, 69(1-2):159–171, 2004.

[40] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 120–130. ACM Press, 2000.

[41] B. Liang and Z. J. Haas. Virtual backbone generation and maintenance for ad hoc network mobility management. In *Proceedings of the 19th IEEE INFOCOM, vol. 3*, pages 1293–1302, Mar 2000.

[42] H. Luo, R. Ramjee, P. Sinha, L. E. Li, and S. Lu. UCAN: A unified cellular and ad-hoc network architecture. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 353–367. ACM Press, 2003.

[43] N. Malpani, J. L. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 96–103. ACM Press, 2000.

[44] A. B. McDonald and T. F. Znat. Mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1466–1487, Aug 1999.

[45] Mesquite Software Inc. CSIM18 simulation engine.

[46] D.-X. Ou, K.-Y. Lam, and D.-C. Dong. An adaptive direction-based location update scheme for next generation PCS networks. In *DEXA '02: Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 413–422. Springer-Verlag, 2002.

[47] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the INFOCOM '97, vol.3*, pages 1405–1413. IEEE Computer Society, 1997.

[48] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, pages 90–100. IEEE Computer Society, 1999.

[49] N. B. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 1–14. ACM Press, 2001.

[50] J. Schiller and A. Voisard. *Location-Based Services*. Morgan Kaufmann, Apr 2004.

[51] A. Y. Seydim, M. H. Dunham, and V. Kumar. Location dependent query processing. In *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 47–53. ACM Press, 2001.

[52] M. Tsukamoto, R. Kadobayashi, and S. Nishio. Strategies for query processing in mobile computing. In T. Imielinski and H. F. Korth, editors, *Mobile Computing*, pages 595– 620. Kluwer Academic Publishers, Boston, 1996.

[53] K. H. Wang and B. Li. Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks. In *Proceedings of the 21st IEEE INFOCOM, vol.2*, pages 1089–1098, Jun 2002.

[54] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.

[55] O. Wolfson, S. Chamberlain, S. Dao, and L. Jiang. Location management in moving objects databases. In *Proceedings of the Second International Workshop on Satellite-based Information Services*, pages 7–13, Oct 1997.

[56] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distributed and Parallel Databases Journal*, 7(3):257–387, Jul 1999.

[57] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. In *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks*, pages 102 – 111. IEEE Computer Society, 2001.

[58] R. Zheng, J. C. Hou, and L. Sha. Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 35–45. ACM Press, 2003.

Appendix

Part of the research results of this thesis has been published in the following referred papers:

G. H. K. Lam, H. V. Leong, and S. C. F. Chan. GBL: Group-based location updating in mobile environment. In *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, pages 762–774, 2004.

G. H. K. Lam, H. V. Leong, and S. C. F. Chan. Reducing group management overhead in group-based location management. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA 2004)*, pages 640–644, 2004.

G. H. K. Lam, H. V. Leong, and S. C. F. Chan. Leadership maintenance in group-based location management scheme. In *Proceedings of OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004, Part 1*, pages 544–562, 2004.

G. H. K. Lam. A Framework for Group-based Location and Data Management in Mobile Environment. In *Proceedings of ACM Postgraduate Research Day*, pages 249–256, 2004.