

國立交通大學

資訊工程學系

碩士論文

MobiServNet - 普及運算中有效率之行動服務發現機制

MobiServNet - An Efficient Mobile Service Discovery for Ambient
Network



研究生：蕭志鵬

指導教授：彭文志 教授

中華民國九十四年十月

MobiServNet – 普及運算中有效率之行動服務發現機制
MobiServNet - An Efficient Mobile Service Discovery for Ambient
Network

研究生：蕭志鵬

Student : Chih-Peng Hsiao

指導教授：彭文志

Advisor : Wen-Chih Peng

國立交通大學
資訊工程學系
碩士論文



A Thesis
Submitted to Department of Computer Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

October 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年十月

MobiServNet - 普及運算中有效率之行動服務發現機制

學生：蕭志鵬

指導教授：彭文志 博士

國立交通大學資訊工程學系（研究所）碩士班

摘 要

由於無線通訊系統越來越普及化，我們被越來越多連接到他們個別網路系統的行動服務裝置所環繞著。這樣的行動服務裝置可能像是車輛、手機與個人數位助理，甚至是各式各樣的行動感測裝置系統。在眾多重要的議題之中，我們專注在如何找尋適當的行動服務。為了支援良好的擴充性和有效率地服務找尋，設置和查詢大量具空間性且具時間性的服務資訊是必需的。在這篇論文中，我們設計一個有效率的服務發現機制，這個機制中使用具空間性且具時間性的服務資訊將資訊分組。這個被提出的機制(稱之為 MobiServNet)探究一個雜湊技術和應用層的位置繞行綱要，用來有效率地發現被要求的服務。此外，服務宣傳和詢問請求繞行方法也提出用來獲得適當大概的服務資訊。實驗模擬結果驗證我們所提出的基於具空間性且具時間性的服務資訊和網路技術可以顯著地改善效能。


MobiServNet – An Efficient Mobile Service Discovery for Ambient Network

student : Chih-Peng Hsiao

Advisors : Dr. Wen-Chih Peng

Department (Institute) of Computer Science
National Chiao Tung University

ABSTRACT



Due to the growing popularity of wireless communication systems, we are surrounded by increasingly mobile service devices that are connected to their own wireless network systems. Such mobile devices as vehicles, cell phones, PDAs, and mobile sensors are widely available to provide mobile services for users. Note that one of the important issues is to search appropriate mobile services. In order to support scalability well, and make the service discovery efficiently, it is necessary to locate and query a large amount of spatial temporal service information. In this paper, we devise an efficient service discovery mechanism in which the workloads are balanced and spatial-temporal information is utilized for grouping. The proposed mechanism (referred to as MobiServNet) explores a hashing technique and application-layer location routing scheme to efficiently discover the services required. Furthermore, service advertisements and queries routing methods are proposed to obtain appropriate approximated services. Performance of the proposed mechanism is comparatively analyzed. It is shown by our simulation results that by exploiting the technique of overlay networks and spatial-temporal information of mobile servers, our proposed mechanism can lead to significant performance improvement.

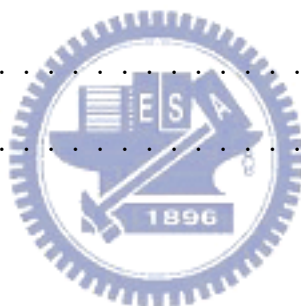
誌 謝

在此要感謝很多人的教導，讓我在研究所兩年中學到許多東西。非常感謝指導教授彭文志博士，兩年來對於我的教導和論文上的指導，讓我受益良多。感謝口試委員：沈錕坤博士及黃俊龍博士對於本論文的指教。感謝在研究所兩年中互相學習、提攜的學長、同學和學弟們，提供給我獲得他們的經驗。另外，也謝謝實驗室的同學和學弟們，給我快樂又充實的兩年。最後，謝謝這多年來支持我的家人和朋友們，讓我無後顧之憂地唸完研究所，沒有你們，我無法成功地完成研究工作。



Contents

1	Introduction	4
2	Preliminary	8
2.1	An Overview of MobiServNet	8
2.2	Components of MobiServNet	10
2.3	Query Semantics of MobiServNet	12
3	Query Processing Mechanism of MobiServNet	14
3.1	Location Data Model	14
3.2	Location Overlay Network	18
3.3	Advertisement Routing	20
3.4	Grouping Services	22
3.5	Query Routing	23
4	Experiments and Analysis	24
4.1	Simulation Model	24
4.2	Performance Measurement	25
4.3	Experiment Results	27
5	Conclusions	30



List of Figures

1	Ambient network. The user might bring another mobile network moving among the house network , train network, and other different wireless networks. . . .	5
2	A traditional mobile objects database.	9
3	An illustrative example of a traditional location range query, a two-dimensional block represent one query, any nodes inside this scope will be reported.	9
4	Six mobile objects and the four feasible MBRs, a dash block represent a MBR.	10
5	The MobiServNet architecture.	11
6	Mobile services moving against time.	13
7	An example for region division.	17
8	An overlay network, where circles represent directories ad dashed line mark the network logically boundary. Right part demonstrates the zone code and boundaries.	19
9	A visual example of the advertisement registering mechanism.	20
10	An example of routing advitiselements.	22
11	An example that query is decomposed and routed to several places.	24
12	Experiment 1: Impact of scalability	27
13	Experiment 2: Impact of update	29
14	Experiment 2: Impact of update.	29
15	Experiment 3: Impact of directory storage	30
16	Impact of retrieval cost	31

List of Tables

1	Query results returned for Q	14
---	--	----



1 Introduction

Due to the growing popularity of wireless communication systems, we are surrounded by increasingly mobile service devices that are connected to their own wireless network systems. Such mobile service devices as vehicles, cell phones, PDAs, and even mobile sensor base stations used to forecast weather. With the advance of recent technology, an increasing number of users are requesting service information via various wireless communications. As shown in Figure 1, at home or outdoors where wireless communications are different, users finding service among different communications is unhandy and absence of collaboration. For example, mobile users want to find some mobile services, but service directory $D1$ and service directory $D2$ might be independent so that they must really know how to request two systems respectively. Therefore, to overcome the aforementioned defects, these communications must be merged into an increasing ubiquitous network infrastructure referred to as ambient networks.

The ambient network is built to integrate the heterogeneity arising from different wired and wireless network technologies such that it appears heterogeneously to the potential users of the network services. The idea of ambient networks has two basic requirements that future networks must satisfy: (1) provide everywhere connectivity of various form, coverage, capacity such as supported by wireless LANs or mobile communication networks (GSM, GPRS, bluetooth, UMTS), and (2), support the users with advanced query features. Once ambient network can satisfy basic requirement, it then should offer an infrastructure of flexible service registration and acquisition mechanisms. Because of the potential huge number mobile devices connected to the ambient network, how to efficiently discover the appropriate location-based data and mobile service providers is very important and needs to be concerned.

Mobile Service Network (referred to as MobiServNet) is a efficient service infrastructure for discovering mobile services offered by mobile service providers in ambient networks. MobiServNet receives advertisements containing location information from mobile service providers

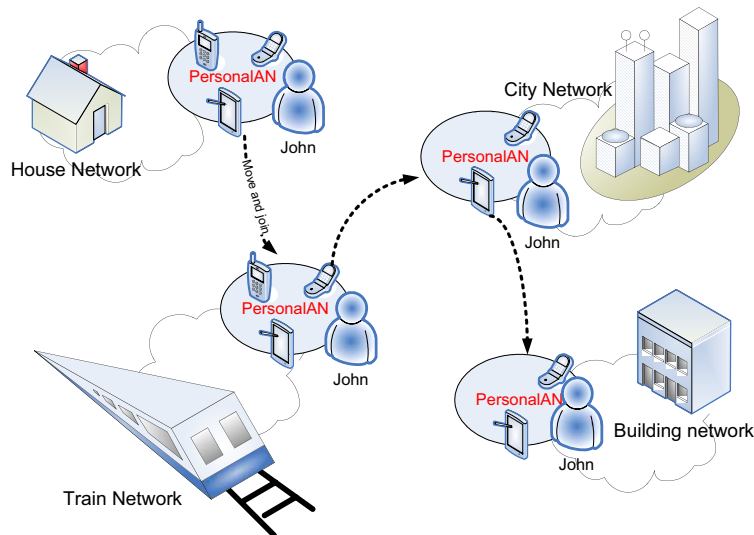


Figure 1: Ambient network. The user might bring another mobile network moving among the house network , train network, and other different wireless networks.

and queries for specified services within a specified zone and specified duration. For each advertisement, MobiServNet adjusts its search structures for the incoming information. For each query, MobiServNet finds the set of mobile services satisfying those given constraints. When needs, MobiServNet also updates the set as mobile service providers move over time. Advertisements contain location information as functions over time. Functions can allow computing position between each two successive position records by mathematical interpolation method. A service provider doesn't have to update its advertisement with position function until the function is no longer reflecting its position information.

In particular, the MobiServNet has the following challenges needed to conquer: (1) it has huge storage cost since mobile service information is usually multidimensional data(i.e., service description data, time, location, etc.); (2) it has high update cost since mobile service equipment might keep moving and have to update location information as time passes; (3) it has high retrieval cost due to the large amount of data; (4) the data need to handle are large-scale since these databases concerned to maintain temporal data (i.e., past, now and predictive future).

A natural way to discover mobile services of interest will be to use spatial-temporal range

queries[3][7][13][12]. Spatial-temporal queries are typically solved in database systems using indexing techniques. The mostly indexing techniques is focused on centralized indices in database systems[6][11], but distributed indexing techniques should be concerned for the scalability.

To achieve scalability, advertisement and query processing in MobiServNet is distributed to a set of directories on an overlay network. Each directory owns zones which dictates the advertisements and queries for which the directories are responsible. MobiServNet directories can be deployed alongside either wireless infrastructure(such as basestations, or gateways), or wired infrastructure. MobiServNet directories function at the application layer, and can work over any underlying physical infrastructure.

To reduce the retrieval cost, queries cannot only specify location and time constraints but also service content constraints. For instance, a practical query with three kind of constraints, find a taxi equipped with an LCD monitor on the Jhongsiao E. road in five minutes. Queries with service content constraints can help users efficiently drill-down their retrieval cost.

To decrease the update cost, we introduce simple temporal and spatial grouping methods. In reality, the moving behavior of mobile objects is usually regular and follows some moving patterns. As some small groups of mobile objects, it can avoid large volume updates sent to the directories . Therefore, we conclude some small behaviors to form groups among the mobile service devices.

To lessen the storage cost, our location information representation differently from traditional position functions is proposed. In traditional position functions, the position data of mobile service devices was represented as a two dimensional geometric coordinate (x,y) and provided by GPS. However, instead, we propose to reveal location information using both administrative zone allocation and road organization combined in one dimensional space that fits better the real-world human interpretation than traditional ones. For instance, if a mobile service provider is moving on a road with a coordinate of latitude=25.047811 and

longitude=121.51264, it can be expressed as a set of fields according to an administrative zone allocation such as city, road-name, road-block (e.g., Taipei City, Jhongsiao E. Rd., 1). Furthermore, by converting the fields into a binary string that has efficient ways to process queries, we also overcome the aforementioned scalability challenge of the ambient network.

To complete the query results, MobiServNet directories route advertisement and queries among themselves. The routing protocols are designed to conserve processing overhead, which can be significant if the protocols are poorly designed. Our work is similar to CAN[9] in that both construct a region-division overlay network[14][5][10] above the underlying physical network. Nevertheless, the two systems are very different in underlying details: the overlay network represented by CAN is purely logical, while our approach is combined contemplation with the under physical and logical topology.

We believe that MobiServNet will be an essential, but perhaps not necessarily the only, distributed structure supporting efficient queries on overlay networks. The proposed solution has more advantages: (1) Both average advertisement and query cost are efficient within a N-directory network (scale as $O(\sqrt{N})$). (2) Since location information is based on the information of the administrative zone allocation, the results can be easily converted into address formats easier for human users to interpret. (3) In the real world, most mobile service equipment can only move following along the "roads". However, if one expresses location information as geometric coordinates, one may include spaces where mobile service equipment can never move into, so called dead space, incurring storage waste. (4) This solution can reduce the storage cost and dimensions of index by revealing location information in one dimensional space, instead of two dimensional space. Also, this results in the improved query processing.

This paper is organized as follows. MobiServNet components and query semantics are represented in Section 2. Distributed processing algorithms developed for MobiServNet are developed in Section 3. Experimental results are presented and analyzed in Section 4. This paper concludes with Section 5.

2 Preliminary

2.1 An Overview of MobiServNet

One major difference between mobile objects and static objects is that the location of moving varies over time. In the database, if we want to store the exact location information of objects, it is inevitable to employ a great volume updates to the database. Therefore, we apply a "fuzzy" view: we do not update the location of objects in database unless it leaves its original position very far.

There is an interesting fact that some uncertainty exists in the query result. In a simple spatial-temporal query Q , given a query range $Q.rect$, the result should include two parts: some objects are definitely in the query range and some may or may not be and need further validation. The uncertainty of the query result causes the idea less useful in cases when exact answers are needed.

This observation supports our fuzzy view that we need not update every little while. In order to solve the discovery problem efficiently, we apply the observation to design a new system structure. Before we present our design, we first review some basic data structures of the other methods. In the traditional structure, as shown in Figure 2, mobile objects send their latest status (such as location, function, velocity etc.) directly to the centralized database. The database then updates the corresponding records. Different index structures (STR-tree[8], TPR-tree) are used in the database to accelerate the update procedures. All queries are answered based on the information stored in database.

Traditional approaches representing any object's location at some time t is often given by $\bar{x}(t) = (x_1(t), x_2(t), \dots, x_d(t))$, where it is assumed that the times t are not before the current time t_{cur} so that the location information can be presented in d -dimensional space at $t \leq t_{cur}$. This position model is revealed as a linear function of time, which is specified by two parameters. The first one is a specific point for the object at some specified time t_{ref}

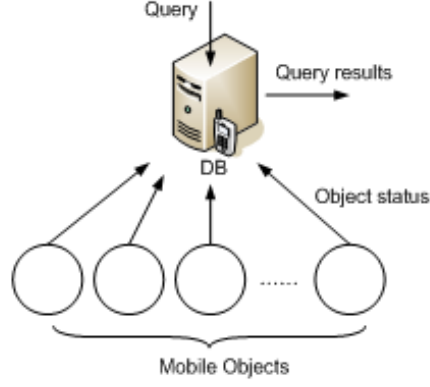


Figure 2: A traditional mobile objects database.

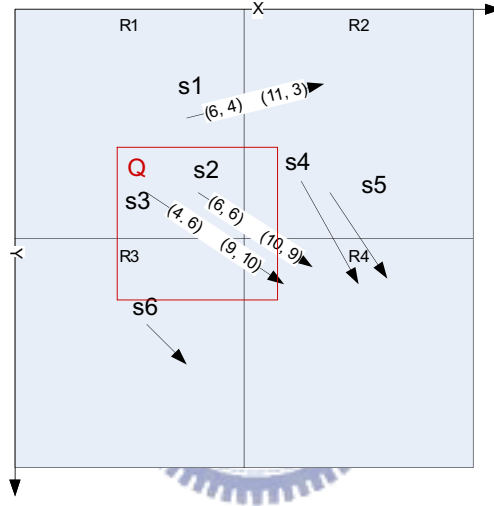


Figure 3: An illustrative example of a traditional location range query, a two-dimensional block represent one query, any nodes inside this scope will be reported.

, $\bar{x}(t_{ref})$, which we term the reference position. The second parameter is a velocity vector for the object, $\bar{v} = (v_1, v_2, \dots, v_d)$ at t_{ref} . Thus, the linear time-position function is given by $\bar{x}(t) = \bar{x}(t_{ref}) + \bar{v} * (t - t_{ref})$.

This position modeling of mobile objects represented as linear functions of time not only enables us to make tentative future predictions easily, but also solves the problem of the incessant updates that would otherwise be required to approximate continuous movement in a traditional setting. For example, objects may report their positions and velocity vectors when their actual positions deviate from what they have previously reported by some threshold. The choice of the update frequency then depends on the type of movement, the desired accuracy,

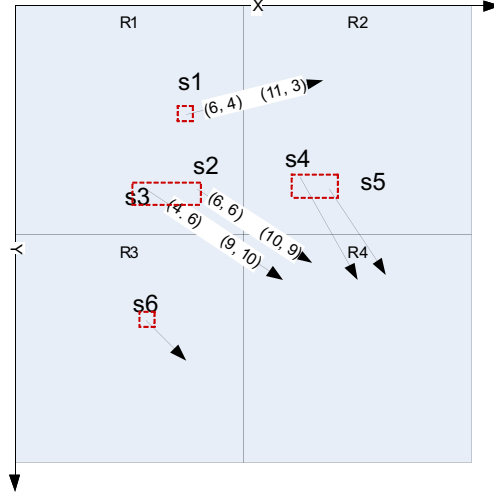


Figure 4: Six mobile objects and the four feasible MBRs, a dash block represent a MBR.

and the technical limitations.

In this fashion, as will be illustrated in the following and explained later, the reference position and the velocity are used not only when recording the future trajectories of moving points, but also for representing the coordinates of the bounding rectangles in the index as functions of time.

Consider an example in Figure 4. The diagram shows the positions and velocity vectors of 7 point objects at time 0. The dash blocks show one possible assignment of the objects to minimum bounding rectangles (MBRs) assuming a maximum of three objects per node. And the indexing solution usually use R-tree[4][2] relevance techniques by taking MBRs as its leaf nodes.

In this paper, we present a service discovery infrastructure (to be referred to as MobiServNet) in which we explore a hashing technique and application-layer location routing scheme to efficiently discover the services required.

2.2 Components of MobiServNet

In MobiServNet, there are three types of major components: location directories that receive both advertisements and queries, service providers that register advertisements, and clients

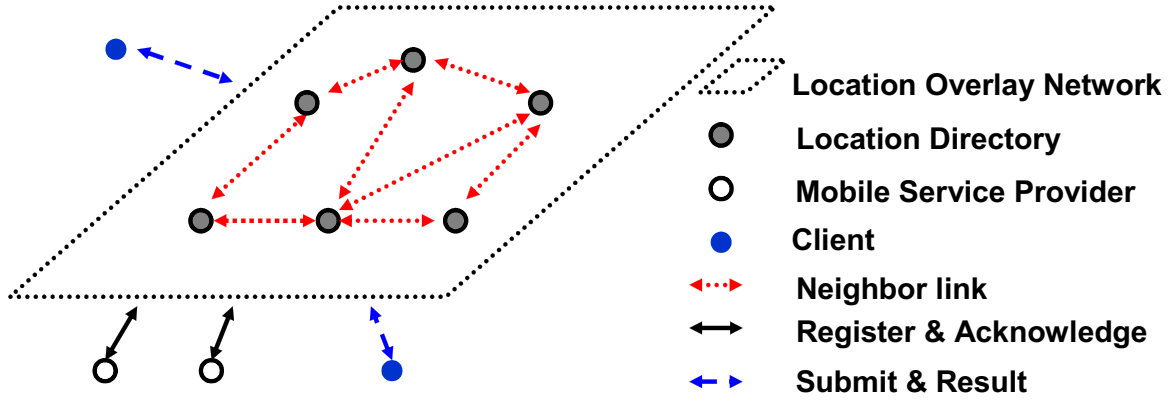


Figure 5: The MobiServNet architecture.

that submit queries.

Location Directories Location directories are located in the ambient network and receive both advertisements and queries. Each one of these directories has at least a pair of a zone code and a zone owner-prefix length. Consider a rectangle R that represent a chosen region on the specified plane. Intuitively, R is the bounding rectangle that contains all directories within the ambient network. We call a sub-rectangle Z of R a *zone*. The zone code $zcode(Z)$ is a 0-1 bit string of length $level(Z)$, and is defined as follows. If Z lies in the *left* half of R , the first (from the left) bit of $zcode(Z)$ is 0, else 1. If Z lies in the *bottom* half of R , the second (from the bottom) bit of $zcode(Z)$ is 0, else 1. The remaining bits of $zcode(Z)$ are then recursively defined on each of the four quadrants of R . The zone code for the location of a directory ld is called $dcode(ld)$. The zone owner-prefix length of a directory ld , $zop(ld)$, is defined to indicate the prefix length of the largest zone which ld primarily owns. More detail content about zones will be discussed later.

Mobile Service Providers Providers notify MobiServNet of their location by registering advertisements to a location directory in the ambient network. Advertisements contain an unique identifier for a service ($SUID$), the mobile service’s network address ($addr(MS)$), an expiration time ($tExp$), location information function (pos), and service content ($keys$). Advertisements ads will be sent to a directory such that the providers resides in the directory’s

responsible zone. The detail how service advertisements find the correct directory as they move throughout the chosen region R will be shown in the following sections. The location information pos takes time as a parameter and returns a pair of reference code and velocity. The reference code $rcode(MS)$ is represented as a binary string concatenated by transformed triplet $(zone, road, position)$. The velocity $v(MS)$ is a vector for representing mobile service moving status. Figure 6 shows an example of location information function for 4 mobile providers, this diagram shows functions in one dimension plotted against time. For a mobile service provider ms that registers an advertisement ad to any directory at time t_{ad}^{Xmit} , ms sends a successive advertisement with new location information function when ms detects that it has moved outside of $ad.pos(t)$ at any time $t: t_{ad}^{Xmit} \leq t \leq ad.tExp$. Discontinuities in Figure 6 indicate changes in the location information functions. If the location information function does not change, ms will probably refresh the advertisement by registering a new one at time $ad.tExp - \Delta$, where Δ is enough time for the advertisement to arrive at the target directory before $ad.tExp$. The service content $keys$ is a filter of a collection of the keyword set that can express mobile service content.

Clients To query MobiServNet, clients submit queries to any attached directory in the MobiServNet network. Each query Q has a rectangle, $rect$, the client's network address $addr(C)$, an expiration time $tExp$, and searching content $keys$. Figure 6 shows a query Q , where the rectangle is only a one-dimensional range of values for a time interval. Q expires at $t = 12$. The query will revoke running after $t = 12$. In the following section, we figure out the semantics of the query result.

2.3 Query Semantics of MobiServNet

When a client submits query Q to any directory at time t_Q^{submit} , the directories would work together to find a set of matching mobile service providers in $Q.rect$. The result set continually

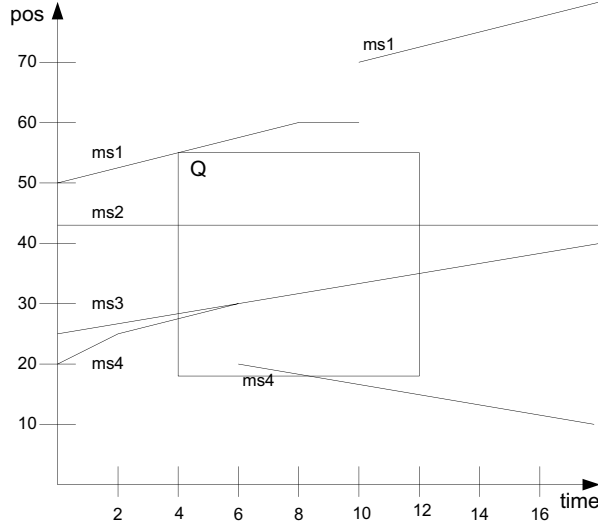


Figure 6: Mobile services moving against time.

changes as these mobile service providers move in and out within $Q.rect$. Consider Q in Figure 6, for example, where $t_Q^{submit} = 4$. At beginning, Q 's result set would contain query responses for $ms2$, $ms3$, and $ms4$. At $t = 12$, the Q 's result set should contain query responses for $ms2$, $ms3$.

As directories receive a new query Q at time $t_Q^{submit} + \Delta$, directories find all cached advertisements ads such that $ad.pos(t)$ fits in $Q.rect$ for some t : $t_Q^{submit} \leq t \leq ad.tExp$ and a matching $ad.keys$ with $Q.keys$. The registration and query acquisition mechanisms we introduce in next section guarantee that queries finds all matching advertisements.

Query Result A query answer qa contains a set of an unique identifier for a service identifier ($SUID$), a start time (t^{Start}), an expiration time ($t^{Expiration}$), the mobile service's network address ($addr(MS)$), and location information function (\overline{pos}) from advertisements ads of location directories in accordance with $Q.rect$ and $Q.keys$ during $t_Q^{submit} \leq t \leq t^{Expiration}$ such that $ad.pos(t)$ resides in $Q.rect$ and $t^{Expiration} = \min(ad.tExpiration, t^{leave})$, where t^{leave} is the smallest time when $ad.pos(t)$ no longer belongs $Q.rect$.

The responsible directories return initial query results to the origin client and incessantly answering the query over subsequent incoming advertisements. Consider an new advertisement

ad_{new} that arrives at the responsible directory where Q is running incessantly. If ad_{new} 's location information function and service summary match Q within the lifetime of the ad_{new} , the directory send a query response qa for Q 's client. Again, routing ensures that all queries find all matching services.

To show the overall query semantics, we refer to the example in Figure 6. To keep our example simple, we assume that any ad transmitted at time t^{Xmit} arriving at a directory at time t has an expiration time of $t^{Xmit} + \Delta + 2 \text{ min}$, where $t^{Xmit} + \Delta \approx t$. We also assume that all ads are examined on even numbered minutes. Consider query Q , with $t_Q^{submit} = 4$ and $Q.t^{Expiration} = 12$. Table 1 shows the query results generated for Q at every examining time.

3 Query Processing Mechanism of MobiServNet

In this section, we show how the MobiServNet architecture supports the above semantics in a distributed environment.



3.1 Location Data Model

As practical respects, representing location of mobile objects as addresses is more intuitive than representing as geographic coordinates. In reality, addresses are usually revealed as multiple-attribute data such as administrative zone name, road name, and position on the road. For example, the address of the National Chiao Tung University, Taiwan could be represented as a triplet of (Hsinchu, Dasyue Rd, 1001). By adding more attributes, it is

Time	Query Results ($t^{Start}, t^{Expiration}, pos$)
4	$\{ms2(4, 6 + \Delta, 43, 0), ms3(4, 6 + \Delta, 28, 1), m4(4, 6 + \Delta, 26, 2)\}$
6	$\{ms2(6, 8 + \Delta, 43, 0), ms3(6, 8 + \Delta, 30, 1), m4(6, 8 + \Delta, 20, -1)\}$
8	$\{ms2(8, 10 + \Delta, 43, 0), ms3(8, 10 + \Delta, 32, 1)\}$
10	$\{ms2(10, 12 + \Delta, 43, 0), ms3(10, 12 + \Delta, 34, 1)\}$

Table 1: Query results returned for Q .

trivial to extend the scheme to be able to support more general addresses. From here forward, to keep the presentation simple, we only focus on the triplet scheme, (zone, road, position on road), to represent addresses within a specific region.

By taking advantage of the addressing schema above, one can easily transform the location of a mobile object into a one-dimensional binary string. The procedure to transform the location of a mobile service device into a binary string consists of four steps as follows: (1) transform the location of the mobile objects given by GPS into a address form (2) obtain the address of the place at which the mobile object is located and represent it as a triplet, (3) transform each attribute of the triplet into a binary string, and (4) concatenate the three binary strings into a single binary string. We assume that the first step can be done by local mechanisms such as administrative mapping. The second and fourth steps are trivial, and thus we detail the third step.

To complete third step, we first introduce how to produce the zones. For easier illustration, consider an imaginary region with 2 countries (A, B) as a whole and 16 cities (0,1,...,9,a, b, ..., f) in each country - a total of 32 zones to transform. The simplest transforming method is to use their lexicographical orders (numbers < letters). That is, by using 1 bits for county names and 4 bits for city names, one can transform a zone into a 5-bit string whose first one bit represent the lexicographical order of its county name and the remaining four bits represent the lexicographical order of its city name. For example, one can express the zone "A county a city" as "0 0110", the zone "A county b city" as "0 0111" and "B county a city" as "1 0110".

Although the simplest transforming scheme is simple to implement, it does not provide the information about relative location of zones. For example, consider two mobile objects, one located at the zone "0 0110" and the other at the zone "0 0111". Comparing these two binary strings, one can deduce that the two objects be in the same county (A) but in different city (a,b). These two binary strings, however, do not provide any clue as to the relative location information of the two objects.

To overcome the limitation that transforming scheme does not provide the information about relative location of zones, using a mapping technique based on space-filling curves is proposed. A space-filling curve[1] is a dimension reduction technique that forms an one-dimensional curve which visits every point within a multi-dimensional space. In order to represent the relative locations of zones more efficiently, we choose bit-shuffling among various space-filling curves and apply it to start region division from the lower left corner as the order in the bit-shuffling. The detailed algorithm to transform the zones contained in a region is given in *Algorithm* region division, and an illustrative example is shown in Figure 7.

A zone is defined by the following constructive procedure. Consider a rectangle R that represent a chosen region on the specified plane. Intuitively, R is the bounding rectangle jointly defined by all directories within the ambient network. We call a sub-rectangle Z of R a *zone*, if Z is obtained by dividing R l times, $l \geq 0$, using a procedure that satisfies the following property:

After the i -th division, $0 \leq i \leq l$, R is partitioned into 2^i equal number rectangles. If i is an odd (even) number, the i -th division is parallel to the y-axis (x-axis).

That's, the bounding rectangle R is first sub-divided into two zones at level 0 by a vertical line that splits R into two equal number of pieces, each of those sub-zones can be split into two zones at level 1 by a horizontal line, and so on. We call the non-negative integer l the level of zone Z , i.e. $level(Z) = l$.

Algorithm: region division

Input: a region R , administrative information of all zones Z

Output: binary strings which zones transformed into, $zcode(Z)$

1. Determine the centroid of each zone, set level $l = 0$
2. **If** $size(R) > 1$
3. \triangleright $size(R)$ is the same with the number of the centroid of zones, because R contains all zones
4. **then** Divide the region R into two sub-regions, west sub-region WR and east sub-region ER
5. $size(WR) = size(ER) = size(R)/2$
6. l -bit of $zcode(WR)$ is 0, l -bit of $zcode(ER)$ is 1, and $l = l + 1$
7. **If** $size(WR) > 1$
8. **then** Divide the region WR into two sub-regions, south sub-region SR and north

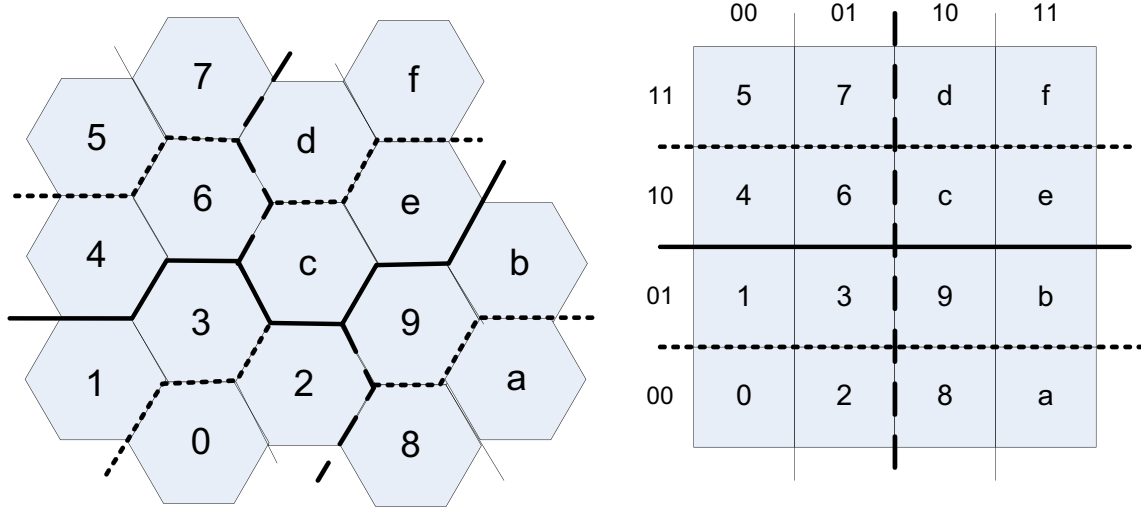


Figure 7: An example for region division.

sub-region NR

9. $size(SR) = size(NR) = size(WR)/2$
10. l -bit of $zcode(SR)$ is 0, l -bit of $zcode(NR)$ is 1, and $l = l + 1$
11. **For** each sub-region WR , ER , SR , and NR obtained from line 2 to line 10
12. **do if** $size(SR) > 1$
13. **then** continue process from line 2 to line 10 to replace R by SR .
14. **If** $size(SR) \leq 1$
15. **then** result the final $zcode(Z)$ of each zone Z .

Compared to the transforming method based on lexicographical orders, the proposed bit-shuffling method produces more informative binary strings. Again, let us consider the two mobile objects, one located at the zone "0 0110" and the other at the zone "0 0111". In addition to the facts that the two objects are in the same county but are in different city, we can attain more clues about the relative location of zones: (1) since the first three bits for city part are both "011", the cities are located at west of southwest area of the county, and (2) since the last bits for city part are different, the city where the first object is located is south of the city where the second object is located.

There are a number of roads within a zone. The algorithm to transform the roads within a zone is not much different from *Algorithm* region division. The changes needed to be made on *Algorithm* region division are as follows: (1) every instance of word "zone" is to be replaced with word "road", and (2) every instance of word "region" is to be replaced with word "zone".

Consider the last part of the location triplet and the way to transform the position on road. We first divide the road into $2^n - 1$ units of the same size, and then represent each boundary as an n -bit binary string. Finally, position of an object on road is chose as the boundary nearest from the object and represented as the boundary binary string.

Intuitively, the suggested transforming scheme has the following characteristics: (1) a location directory owning a set of zones can be represented by the range of binary strings. For example, a directory owning county "A" is represented by the range [00000; 01111], and (2) more shortly, the directory can be revealed by using the longest common prefix bits of the given range.

3.2 Location Overlay Network

From description above, location information of each object can easily transform geographic position into a binary string. Having these binary strings can construct a one-dimensional space that we use to form the location overlay network for supporting the ambient network function. Location overlay network is built atop of physical network layer that is an application layer protocol. Now we describe how zones are mapped to directories.

The location space is logically divided into zones and each zones is assigned to location directories, then it is easy to see that there exists a l such that each directory maps into a independent level- l zone. In real world, the directory deployment are likely to be unnecessarily regular (*i.e.*, grid-like). Some zones may be empty and other zones might have more than one directory situated within them. As applying a simply strategy, the empty zone associates the nearest directory with it for some definition of *nearest*. In order to make overall acquisition mechanism simpler, we allow directories in MobiSerNet to map to different-level zones.

To precisely understand the associations between zones and directories, we give the notion of zone ownership. Consider a location directory ld , let Z_{ld} to be the largest zone that includes only one directory ld . Then, we say that ld owns Z_{ld} . Unfortunately, that ownership decision

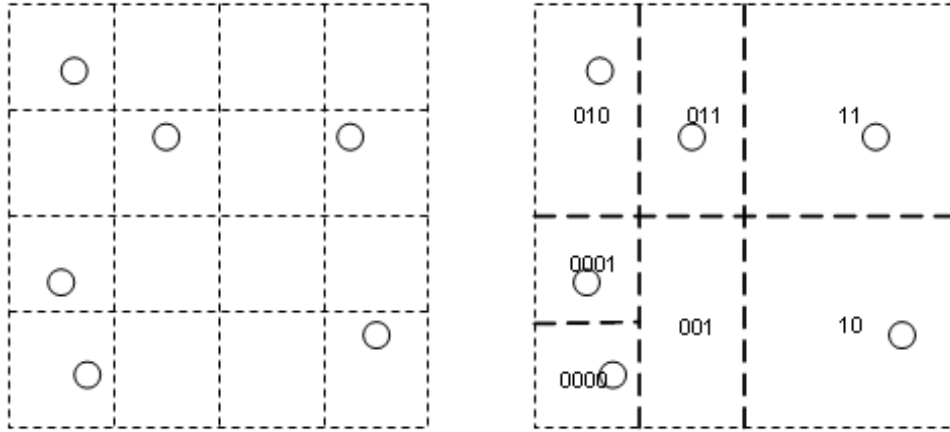


Figure 8: An overlay network, where circles represent directories and dashed lines mark the network logical boundary. Right part demonstrates the zone code and boundaries.

might result in some zones may be empty (no owners associated with them). For example, in Figure 8, the zone 001 does not contain any directories and would not have an owner. To resolve this, for any empty zone Z_{empty} , we take an definition of $nearest(Z_{empty})$ and define the owner of Z_{empty} to be the owner of $nearest(Z_{empty})$. In the example figure, the directory owns 0001 would also own the empty zone 001. For convenience, we still use two dimension representation in our example.

Having defined the relationship between zones and directories, the next problem we address is: how do we develop such a distributed algorithm that enables each directory to determine which zones it owns, knowing only the overall boundary of overlay network? In principle, this should be relatively straightforward, since each directory can simply determine the location of its neighbors, and apply simple methods to determine the largest zone around it such that no other directory resides in that zone.

The proposed distributed zone construction algorithm defers the decision of such zones until when either a query is initiated, or when a advertisement is registered. The basic ideal behind the algorithm is that each directory tentatively builds up an idea of zone it resides in just by negotiating with its neighbors.

Algorithm: zone ownership construction

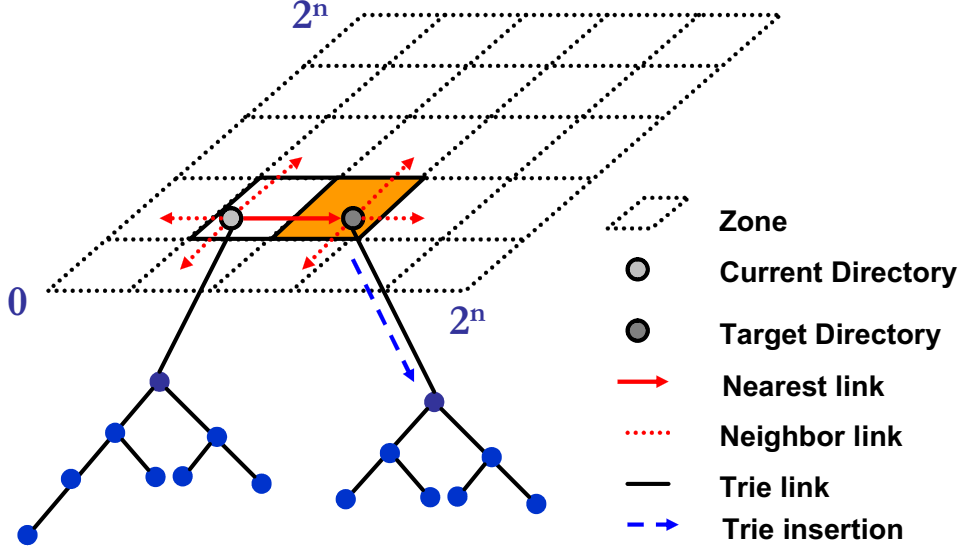


Figure 9: A visual example of the advertisement registering mechanism.

Input: directory codes and zone owner prefixes of a location directory ld owns, and newly discover neighbor directory nd of ld

Output: updated zone owner-prefix length $zop(ld)$ and $zop(nd)$

1. $bitcmp \leftarrow dcode(ld) \text{ XOR } dcode(nd)$
2. **if** $bitcmp = 0$
3. **then if** $zop(ld) \leq zop(nd)$
4. **then** $zop(ld) \leftarrow zop(nd)$, treat ld as slave directory
5. **else** $zop(nd) \leftarrow zop(ld)$, treat nd as slave directory
6. **else if** $zop(ld) \leq zop(nd)$
7. **then for** $l \leftarrow zop(ld)$ **to** $zop(nd)$
8. **do if** $bitcmp[zop(nd)] = 0$
9. **then** left shift $bitcmp$ 1 bit
10. $l \leftarrow l + 1$
11. $zop(ld) \leftarrow l$
12. **else for** $l \leftarrow zop(nd)$ **to** $zop(ld)$
13. **do if** $bitcmp[zop(ld)] = 0$
14. **then** left shift $bitcmp$ 1 bit
15. $l \leftarrow l + 1$
16. $zop(nd) \leftarrow l$

3.3 Advertisement Routing

In this section, we describe how advertisements are registered into MobiServNet. There are two algorithms of interest: a hashing technique for mapping an advertisement to a zone, and a routing algorithm for storing the advertisement at the directory of the appropriate zone.

Consider a mobile service provider that wants to send an advertisement ad to MobiServNet.

Firstly, the ad should be calculated the service summary including the reference code it belongs to. After transforming advertisement ad , to route ad , the attached directory ld first check whether the location information of ad is within the zones ld owns. If ld owns the zone which ad is located and four direction neighbors is determined, store the advertisement ad at the directory ld . If not, check all neighbors of ld to determine the nearest neighbor nld from ad , and then forwarding the ad to nld . Continuing the process above, directory nld will attempt to route the ad to its owner.

More precisely, an advertisements includes an unique identifier for the mobile service ($SUID$), the mobile service's network address ($addr(MS)$), an expiration time ($tExp$), a velocity $v(MS)$, and a service summary that contains a reference code $rcode(MS)$ that the provider belongs to and a hashing binary string $keys$ of service content. The reference code apply the producing method that had discussed in section 4.1. The hashing binary string of service content apply the bloom filter to be produced based on its keywords. The aim of transforming an advertisement to a service summary is to store the advertisement at the directory that owns the zone advertisement belongs to.

Attached directory ld checks the service summary ad by comparing the same length prefix of zone code $zcode(ld)$ with $rcode(ad)$. If the two are identical, directory ld store the ad by its locally Patricia tries structure ,as shown in Figure 9. If not, ld forwards ad to the destination-closest neighbor directory nld . Destination-closest neighbor directory nld is decided by applying right-hand rule that is clockwise first. For example, as shown in Figure 10, an advertisement ad with $rcode(ad)$ "0110" has arrived zone "10". The first two bits between "10" and "0110" is different so that zone "10" directory forwards ad to its destination-closest neighbor directory which owns zone "110". Repeatedly, the first three bits between "110" and "0110" is different so that zone "110" directory forwards ad to its destination-closest neighbor directory which owns zone "0110". Finally, both four bits are identical, and four direction neighbors of zone "0110" is determined so that the directory which owns zone "0110" will

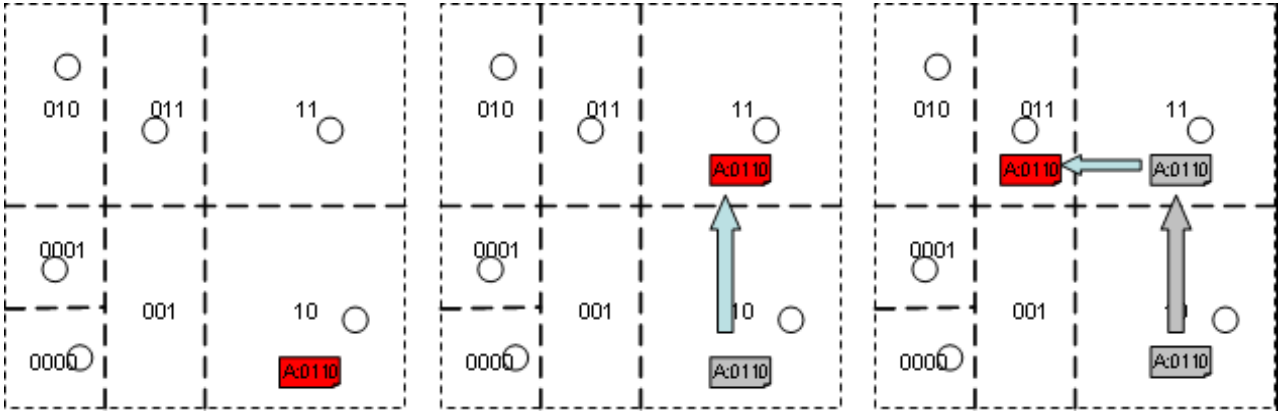


Figure 10: An example of routing advertisements.

store the ad to its local structure.

When the advertisement ad has already arrived the correspondence directory that owns it, the directory will store the ad as a data node of its search structure. Patricia tries that we propose is the search structure resided in the directory. Each data node is represented with last two parts of the $r\text{code}(ad)$, road and position on road.



3.4 Grouping Services

In reality, the moving behavior of mobile objects is usually regular and follows some moving patterns. Grouping services refers to the situation where some common mobile objects tend to have moving together. For example, a group of visitors with the same guide usually has similar movement behavior, the same as the situation that cars on roads with the same direction, near position, and close speed. Although a number of research works have been elaborated upon the impact of group mobility. But they are all designed for ad-hoc networks. In our proposed method, we can easily group these mobile services on the road in a simple way.

1. on the same road , near position, close velocity, is a group
2. the group should be the same direction temporally
3. group can be easily change among roads, zones totally.

3.5 Query Routing

In this section, we describe how queries acquire query answers from MobiServNet. To query MobiServNet, clients submit queries to any attached directory in the MobiServNet network.

Consider a client that wants to send an new query Q to MobiServNet. Firstly, the Q should be calculated the query summary including the zone code of zones it wants to acquire. After transforming query Q , to route Q , the attached directory ld first check whether the location information of Q is within the zones ld owns. If ld owns the zone which Q is requested and four direction neighbors is determined, retrieve the matching advertisement ads at the directory ld and produce a query answer of the ads . If not, check all neighbors of ld to determine the nearest neighbor nld from Q , and then forwarding the Q to nld . Continuing the process above, directory nld will attempt to route the Q to its consulters.

The query summary calculation is almost the same with advertisement summary calculation. Differently, a query Q may have a range rectangle specified consulting not only one directory. Therefore, when first attached directory receives such a query Q with range rectangle $rect$, it will also calculate the common root directories $rlds$ that might several parts composed for query Q . For example, as shown in Figure 11, a query range rectangle can transform to virtual root zones "011" and "11". When the first matching directories receive the sub-queries, they could infer that they should transfer the sub-queries to other responsible directories which has not been received queries.

When the query Q has already been received by the correspondence directories within the query range, the directory will examine its own search structure. Patricia tries that we propose is the search structure resided in the directory. Each data node is represented with last two parts of the $r\text{code}(ad)$, road and position on road. Intuitively, examine the Patricia tries and match the service content.

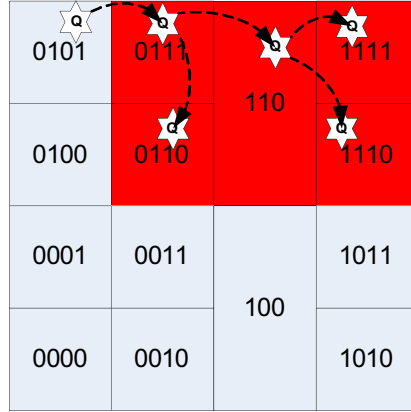


Figure 11: An example that query is decomposed and routed to several places.

4 Experiments and Analysis

In Section 4.1, we describe our simulation model. In Section 4.2, an analytic performance measurements are presented. Section 4.3 is devoted to experimental results.

4.1 Simulation Model

The dataset consists of mobile objects in whole location space which is a unit space $[0, 1]^B$. We use the Java programming language in our experiments due to its strong thread process support. The most other methods are impossible to handle huge number of mobile objects since they index the history information of each object. So it is unfair to compare our method with them. Using this implementation, we conducted a fairly evaluation of MobiServNet and flooding. For all our experiments, we use uniformly directory placement with location network sizes ranging from 50 directories to 300 directories. Each location directory owns a zone. We have conducted experiments at other node densities; they are in agreement with the results presented here.

In all our experiments, each directory initially generates 400 advertisements on average (more precisely, for a topology of size N , we have $400N$ advertisements, and each directories is equally likely to generate an advertisement). We have conducted experiments for different ad-

advertisement distributions. (1) the uniform advertisement distribution generates 2-dimensional locations and, for each dimension, every attribute value is equally likely. (2) the normal advertisement distribution generates 2-dimensional locations and, for each dimension, the attribute value is normally distributed with a mean corresponding to the mid-point of the attribute value range. The normal advertisement distribution represents a skewed data set.

Having generated the dataset, for each simulation we generate queries such that, on average, each directory generates 2 queries. The query sizes are determined using the three size distributions we discussed earlier: uniform, bounded-uniform, and exponential. Once a query size has been determined, the location of the query is uniformly distributed. The later analysis will give us some insight into the asymptotic behavior of various distributions for our primary metrics.

4.2 Performance Measurement

Our performance measurements are described as follows:

Average Advertisement Cost measures the average number of messages required to send the advertisement to the responsible directory. In a N -directory network, the average advertisement cost is proportional to \sqrt{N} .

Average Query Cost measures the average number of messages required to route the query to the relevant directories. The query cost depends upon the range size specified in the query. Consider that our query routing mechanism is careful about splitting a query into sub-queries, doing so only when the query nears the zone that intersects the query range. Thus, the query cost is composed of two components. The first part, which is proportional to \sqrt{N} , is the cost to near the intersecting zone. If the query range can be split into M directories, the second part of query cost is proportional to M . Again, the average query cost depends upon the distribution of query range sizes. Now, suppose that query range sizes follow some density function $f(m)$, then the average query cost can be approximated by $\int_1^N mf(m)dm$.

Consider several different distribution forms: the uniform distribution, the bounded uniform distribution, and the exponential distribution. In all the analysis, we make the simplifying assumption that the size of a query is proportional to the number of directories that can answer that query.

For the uniform distribution $P(m) \propto c$ for some constant c . Thus, the average query cost of uniformly distributed queries is $O(N)$. For uniformly distributed queries, the performance is comparable to that of flooding. However, for the application we envisage, where the directories are responsible to the advertisements of mobile objects, the uniform distribution is highly unrealistic.

More realistically, all query range sizes are bounded by a constant c . In this case, the average query cost is approximated by $\int_1^c mf(m)dm = O(c)$. Thus, the average cost of limited queries is $O(\sqrt{N})$.

Finally, for the exponential distribution, $f(m) = ce^{-cm}$ for some constant c , and the average cost is $O(1)$ for large enough N . Asymptotically, the cost of the exponentially distributed query is dominated by the first part $O(\sqrt{N})$.

Thus, we see that if the queries follow either the bounded uniform distribution or the exponential distribution, the query cost scales as the average advertisement cost.

It does not measure the number of answered messages; this number depends upon the precise data distribution and is at worst the same with the query cost.

Storage Cost measures the total size of stored service summary data. At first the collected data are stored in 3-d R-tree as a triplet (x-coordinate, y-coordinate, timestamp) and then stored in binary Patricia tries as a pair (binary string representation of location, timestamp). Storage cost of data stored in 3-d R-tree is $2 * size(coordinate) + size(timestamp)$, and the storage cost of binary Patricia tries is at worst the same with $B = size(\lceil \lg(\#_of_zones) \rceil) + size(\lceil \lg(\#_of_roads) \rceil) + size(\lceil \lg(positions_on_road) \rceil) + size(timestamp)$.

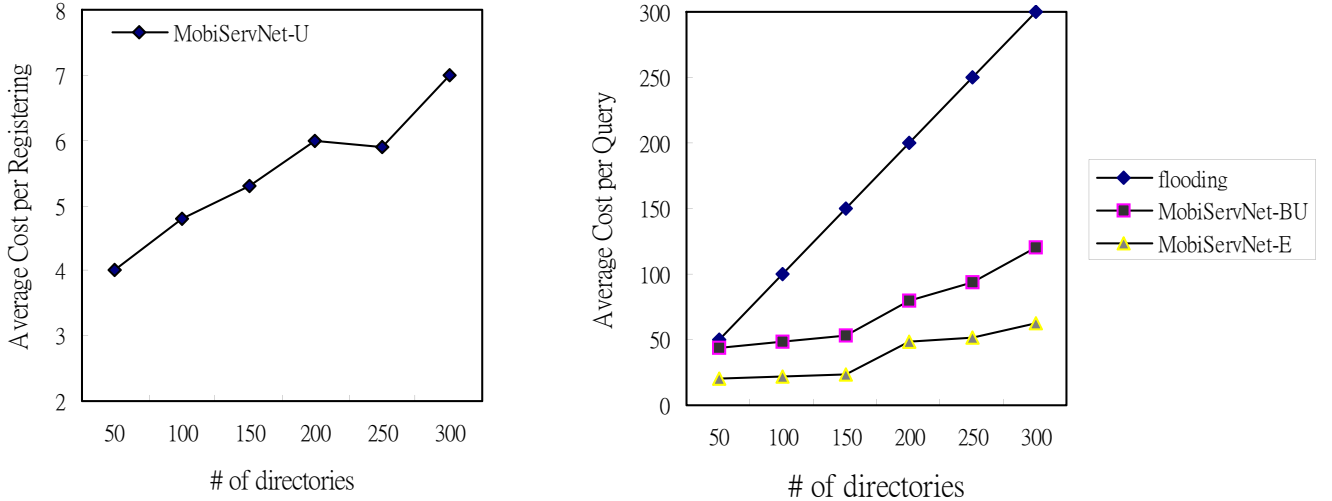


Figure 12: Experiment 1: Impact of scalability

4.3 Experiment Results

Experiment 1: Impact of Scalability

In this section, we use simulation to compare MobiServNet against flooding. Although we have examined almost all the combinations of factors described above, we discuss only the most salient ones here, for lack of space. As shown in Figure 12, the left diagram plots the average advertisement costs for MobiServNet (for flooding, of course, the insertion costs are zero). MobiServNet incurs less per event overhead in registering advertisements. The right diagram plots the average query cost for flooding, the bounded uniform query size distribution, and the exponential distribution. For this graph we use a uniform advertisement distribution, since the advertisement distribution does not affect the query delivery cost. For this simulation, the bound of the bounded uniform distribution was $\frac{1}{4}th$ the size of the largest possible query. MobiServNet-U is MobiServNet with uniform advertisement distribution. MobiServNet-BU and MobiServNet-E mean MobiServNet with bounded uniform and exponential query range distribution respectively. Even for this generous query range size, MobiServNet perform quite well (almost a third the cost of flooding when $N=300$). The average query size for the ex-

ponential distribution was set to be $\frac{1}{16}$ th the largest possible query. The superior scaling of MobiServNet is evident in these graphs. Clearly, this is the vision that might expect MobiServNet to perform best, when most of the queries are small and relatively rare queries are large.

Experiment 2: Impact of Update

In this experiment, we study the performance under different average speed of all objects v .

When the temporal and spatial groups are formed, the group representative will cache the information of recently member positions. The group member does not perform recently update to the responsible directory if streams of updates still belong to the same group range (we referred as a group hit). In fact, streams of updates will belong to the same group with high probability. But the simulation use an approach based on the random waypoint mobility model[15]. Surely, the implementation of the latter one is more easier than the former one.

Mobile object update their advertisements with positions every 125 milliseconds. A position consists of two attributes: an x-coordinate and a y-coordinate. This simulation runs for 300 seconds. The positions of mobile objects vary within $[0,1]^{24}$ virtual space. As shown in the Figure 13, we see that this update mechanism significantly keeps good hit ratio within the groups (by up to 95% with 12,000 nodes). In the Figure 14, MobiServNet-R means MobiServNet whose updates are produced when positions are changed. MobiServNet-G means MobiServNet whose updates are produced when group memberships are changed or position of the group representative changed. Significantly, we see that this group update mechanism reduces the total number of updates to the responsible directories (by up to 45% with 12,000 nodes, compare MobiServNet-G to MobiServNet-R).

Experiment 3: Impact of Directory Storage

While increasing the number of mobile objects from 2,000 to 12,000 (varies number of directories from 50 to 300), we measure the storage size of the 3-d R-tree and binary Patricia

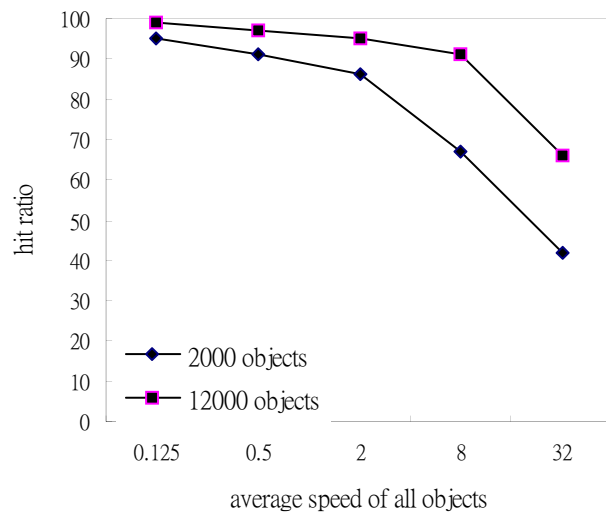


Figure 13: Experiment 2: Impact of update

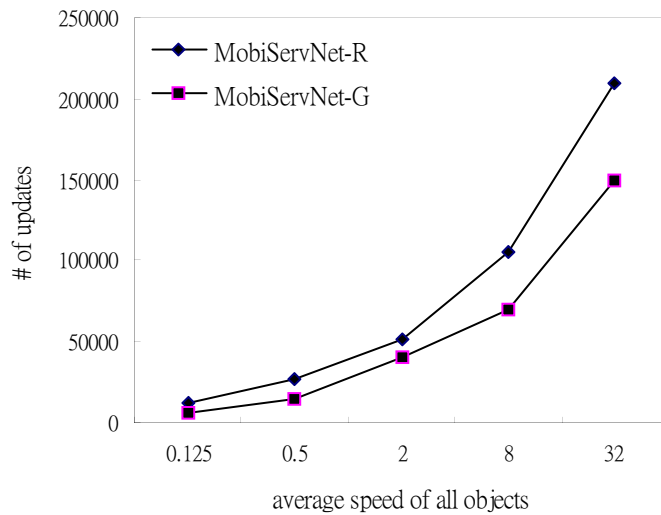


Figure 14: Experiment 2: Impact of update.

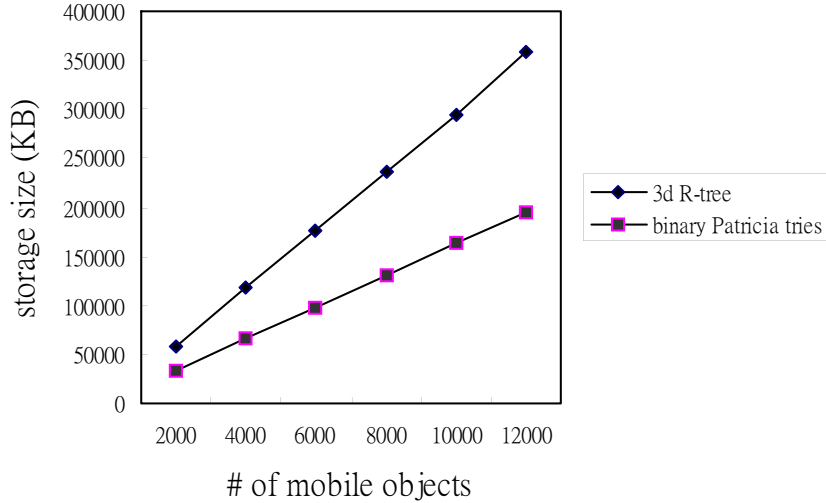


Figure 15: Experiment 3: Impact of directory storage

tries. As shown in Figure 15, the binary Patricia tries (BPT) which stores the locations in binary string representation consumes about 56% of the storage space spent by the 3-d R-tree. Therefore, the reduction ratio of storage space was approximately 44%.

Experiment 4: Impact of Retrieval Cost

Figure 16 plots the number of query results received for different query range sizes. It appears that we received fewer results than expected since the reason we have explained in the analysis. MobiServNet-NK is MobiServNet without keyword constraints, and another one with keyword constraints.

5 Conclusions

In this paper, we devised an effective service infrastructure for discovering mobile services. Our location data model captures mobile objects as binary strings in one dimensional space instead of conventional (x, y) geographic coordinates in two dimensional space that are indexed by R-tree relevance techniques, and thus can reduce storage cost while improving query processing. Our temporal and spatial grouping method provides a better update mechanism

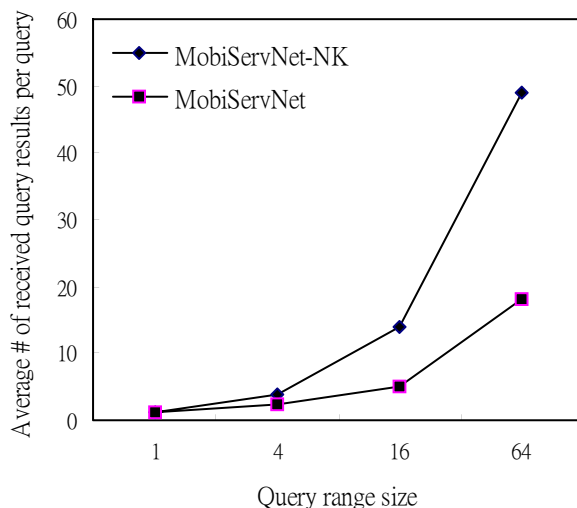


Figure 16: Impact of retrieval cost

and improves the query performance. Our appropriate approximated query results provides the query performance better than all possible results. The design of MobiServNet is fully distributed and more scalable. The simulation studies confirm all above declarations. We also have planed to conduct larger scale experiments in the future.

References

- [1] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmayer. Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science*, 181(1):3–15, 1997.
- [2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, 1990.
- [3] J. L. Bentley and J. H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, 1979.
- [4] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM Press.
- [5] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica. Complex queries in dht-based peer-to-peer networks, 2002.
- [6] G. Kollios, D. Gunopulos, and V. J. Tsotras. On indexing mobile objects. In *PODS*, pages 261–272. ACM Press, 1999.
- [7] H. P. Kriegel, M. Schiwietz, R. Schneider, and B. Seeger. Performance comparison of point and spatial access methods. In *SSD '90: Proceedings of the first symposium on Design and implementation of large spatial databases*, pages 89–114, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

- [8] D. Pfoser. Indexing the trajectories of moving objects. *IEEE Data Eng. Bull.*, 25(2):3–9, 2002.
- [9] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.
- [10] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–??, 2001.
- [11] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 331–342. ACM, 2000.
- [12] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
- [13] B. Seeger and H.-P. Kriegel. Techniques for design and implementation of efficient spatial access methods. In *VLDB '88: Proceedings of the 14th International Conference on Very Large Data Bases*, pages 360–371, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.
- [14] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.
- [15] K. Wang and B. Li. Group mobility and partition prediction in wireless ad-hoc networks, 2002.

