# SCALING LOCATION-BASED SERVICES WITH LOCATION PRIVACY CONSTRAINTS: ARCHITECTURE AND ALGORITHMS

A Thesis
Presented to
The Academic Faculty

by

Bhuvan Bamba

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2010

# SCALING LOCATION-BASED SERVICES WITH LOCATION PRIVACY CONSTRAINTS: ARCHITECTURE AND ALGORITHMS

Approved by:


Dr. Ling Liu, Committee Chair
School of Computer Science
*Georgia Institute of Technology*

Dr. Min Luo
IBM SWG
*IBM*

Dr. Mustaque Ahamad
School of Computer Science
*Georgia Institute of Technology*

Dr. Evaggelia Pitoura
Computer Science Department
*University of Ioannina*

Dr. Douglas M. Blough
School of Electrical and Computer Engineering
*Georgia Institute of Technology*

Dr. Calton Pu
School of Computer Science
*Georgia Institute of Technology*


Date Approved: June 28, 2010

*To all the people who make my world a better place for one reason or another.*

# ACKNOWLEDGEMENTS

I gratefully acknowledge the support and encouragement provided by my advisor, Ling Liu, during the time I spent at Georgia Tech. She has been a most valued friend and mentor during tough times. I appreciate her contribution towards my growth as a researcher and as an individual over the past few years. Her undying enthusiasm and unwavering dedication to her work is an example I wish to follow in my career.

I would like to acknowledge the feedback received from members of my thesis committe: Mustaque Ahamad, Douglas Blough, Evaggelia Pitoura, Min Luo and Calton Pu. I also wish to gratefully acknowledge the financial support Doug provided for me under the MedVault project. Mustaque and Doug also provided me an opportunity to work with people outside my dissertation research area which helped broaden my knowledge scope. I would like to thank them for the short but interesting discussions on privacy related topics which helped me look at my work in a much broader context. Special thanks are also due to Calton Pu for thought-provoking conversations on privacy related topics.

The friendship, companionship and support of my colleagues in the DiSL research group, Systems lab and Databases lab would be hard to replace. Special thanks to Qinyi Wu and Danesh Irani for always being around in the lab. A big thanks to all my collaborators at IBM IRL, IBM T.J. Watson Research Center, IBM Almaden Research Center as well as NTT Docomo Labs. Special thanks are due to Mukesh Mohania, Sougata Mukherjea and Prasan Roy from IRL, Philip S. Yu, Kun-Lung Wu, Bugra Gedik and Arun Iyengar at T.J. Watson Research Center and Madhukar Korupolu and Aameek Singh at IBM Almaden Research Center.

A special thanks to Anupam and Ankit for being such great friends. Most importantly, I would like to acknowledge the unconditional love and support of my parents, Neelam and

Ravi Bamba, and my sister, Rasika Bamba. The more I discover about this world, the more proud I feel of the values my parents have inculcated in me. Thanks to my grandparents for all their love, affection and wisdom; I will always miss you.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Advances in sensing and positioning technology, fueled by wide deployment of wireless networks, have made many devices location-aware. These emerging technologies have enabled a new class of applications, known as Location-Based Services (LBS), offering both new business opportunities and a wide array of new quality of life enhancing services. One example of such services is spatial alarms, an enabling technology for location-based advertisement, location-based alerts or reminders and a host of other applications. On the other hand, the ability to locate mobile users accurately also opens door for new threats - the intrusion of location privacy. The time series of location data can be linked to personal identity, which leads to unauthorized information exposure about the individual's medical conditions, alternative lifestyles, unpopular political views or location-based spam and stalking. Thus, there are two important challenges for location-based service provisioning. How do we scale LBSs in the presence of client mobility and location dependent constraints for the multitude of new, upcoming location-based applications under a common framework? How do we provide anonymous location-based services with acceptable performance and quantifiable privacy protection in the next generation of mobile networks, systems and applications? This dissertation delivers technical solutions to address these important challenges.

First, we introduce spatial alarms as the basic primitive to represent a class of location-based services that require location-based trigger capability. Similar to time-based alarms, spatial alarms serve as spatial event reminders that enable us to express different location-based information needs supported by a variety of applications ranging from location-based advertisements, location-based personal assistants, to friend locator services like Google Latitude. We develop a generalized framework and a suite of optimization techniques for

server-centric scalable processing of spatial alarms. Our architecture and algorithm development provide significant performance enhancement in terms of system scalability compared to naïve spatial alarm processing techniques, while maintaining high accuracy for spatial alarm processing on the server side and reduced communication costs and energy consumption on the client side. Concretely, we develop safe period optimizations for alarm processing and introduce spatial alarm grouping techniques to further reduce the unnecessary safe period computation costs. In addition, we introduce a distributed alarm processing architecture that advocates the partitioning of the alarm processing load among the server and the relevant mobile clients to reduce the server load and minimize the client-to-server communication cost through intelligent distribution and parallelization. We also explore a variety of optimization opportunities such as incorporating non-spatial constraints into the location-based information monitoring problem and utilizing efficient indexing methods such as bitmap indexing to further enhance the performance and scalability of spatial alarm processing in the presence of mobility hotspots and skewed spatial alarm distributions.

Second, we develop the PrivacyGrid framework for privacy-enhanced location service provisioning, focusing on providing customizable and personalized location privacy solutions while scaling the mobile systems and services to a large number of mobile users and a large number of service requests. The PrivacyGrid approach has three unique characteristics. First, we develop a three-tier architecture for scaling anonymous information delivery in a mobile environment while preserving customizable location privacy. Second, we develop a suite of fast, dynamic location cloaking algorithms. It is known that incorporation of privacy protection measures may lead to an inherent conflict between the level of privacy and the quality of services (QoS) provided by the location-based services. Our location cloaking algorithms can scale to higher levels of location anonymity while achieving a good balance between location privacy and QoS. Last but not the least; we develop two types of location anonymization models under the PrivacyGrid architecture, one provides the random way point mobility model based location cloaking solution, and

the other provides a road network-based location privacy model powered by both location $k$-anonymity and segment $s$-anonymity. A set of graph-based location cloaking algorithms are developed, under the MobiCloak approach, to provide desired levels of privacy protection for users traveling on a road network through scalable processing of anonymous location services.

This dissertation, to the best of our knowledge, is the first one that presents a systematic approach to the design and development of the spatial alarm processing framework and various optimization techniques. The concept of spatial alarms and the scaling techniques developed in this dissertation can serve as building blocks for many existing and emerging location-based and presence based information and computing services and applications. The second unique contribution made in this dissertation is its development of the PrivacyGrid architecture for scaling anonymous location based services under the random waypoint mobility model and its extension of the PrivacyGrid architecture through introducing the MobiCloak road-network based location cloaking algorithms with reciprocity support for spatially constrained network mobility model. Another unique feature of the PrivacyGrid and MobiCloak development is its ability to protect location privacy of mobile users while maintaining the end-to-end QoS for location-based service provisioning in the presence of dynamic and personalized privacy constraints.

# CHAPTER I

# INTRODUCTION

The availability of a large number of location sensing technologies ranging from cell network-based positioning of mobile users to highly accurate GPS navigation devices has made location information ubiquitous. This, in turn, has led to the successful deployment of a large number of location-based services and applications. Services range from traffic monitoring and emergency services to location-based advertising and entertainment as well as location-based queries and spatial alarms [25]. The plethora of services available today provide a huge scope for business opportunities. According to research reports [2], LBS revenue is forecast to reach an annual global total of $13.3 billion by 2013. This is further supported by current trends for GPS markets which are growing at 20% annually. Currently, 90% of available GPS devices are portable navigation devices and by 2012 mobile phones equipped with GPS will have 78% of this market [14]. Cell phone users can be accurately located at any point of time using triangulation performed by the network of more than 220,000 cell phone towers [69]. Skyhook wireless lists a database of more than 6.5 million Wi-Fi access points [61], all of which provide the means to locate mobile users. With the advent of Android [11], iPhone [12], Palm Pre [16] and various other smart phones with embedded GPS devices, it is easy to gather accurate location information for a large number of mobile users.

We develop spatial alarms as an interesting example of location-based services, an enabling technology for location-based advertisement and location-based alerts or reminders. Similar to time-based alarms, spatial alarms serve as spatial event reminders that enable us to express different location-based information needs currently required by a variety

of applications ranging from location-based advertisements, location-based personal assistants, to friend locator services like Google Latitude [18]. Spatial alarm processing requires meeting two demanding objectives: high accuracy, which ensures zero or very low alarm misses, and high scalability, which requires highly efficient and optimal processing of spatial alarms. These technical challenges are discussed in detail in Section 1.2.

In order to manage the scalability challenge in location-based systems, we also develop a *location-centric framework* that advocates clean separation of static location data from moving objects in terms of both indexing structure and location query processing. Employing a separate index for moving objects enables us to minimize the index maintenance cost in the presence of location updates of moving objects, and speeds up the evaluation of location queries by maximizing the amount of parallel processing, specifically processing location queries over static objects via location index and processing location queries over moving objects through grid index. We also show that this framework enables dynamic composition of a world wide location index which can support indexing for all locations of interest in the entire world in a hierarchical manner.

On the other hand, the availability of continuous, precise location information to facilitate location-based applications opens the door for potential misuse of private location information of mobile users [64]. The collected location information can be used for stalking [1], perform inference about a user's personal lifestyle habits or medical conditions, or to spam users with unwanted location-based advertising. Location privacy refers to the capability of enabling a mobile node or a trusted location server to conceal the relation between location and personal identifiable information from third parties. In this dissertation, we present a framework for performing personalized anonymization of location information through customizable privacy profiles.

## *1.1 Location-based Systems and Services*

There are two important challenges for location-based service provisioning: how to scale LBSs in the presence of client mobility and location dependent constraints, and how to provide anonymous location services with acceptable performance and privacy protection in the next generation of mobile networks, systems and applications. In this section, we briefly discuss spatial alarms as a generalized, scalable framework for supporting the next generation of location-based services and applications and the inherent problems associated with the availability of continuous location information of mobile users.

### 1.1.1 Spatial Alarm Framework

A spatial alarm is defined by three elements: a future location reference known as the alarm target, an owner who is the publisher of the alarm and the list of subscribers of the alarm. We categorize spatial alarms based on two criteria: the publish-subscribe scope of the alarms and the motion characteristics of alarm targets and alarm subscribers. According to the publish-subscribe scope of spatial alarms, we consider three categories of alarms: *private*, *shared* and *public*. *Private* alarms are installed and used exclusively by the publisher. *Shared* alarms are installed by the publisher with a list of authorized subscribers and the publisher is typically one of the subscribers. *Public* alarms are usually installed with the purpose of sharing them with all mobile users who are entering the spatial regions of the alarms. Mobile users may subscribe to public alarms by topic categories or keywords, such as *"traffic information on highway 85 North"* or *"Zagat survey of top-ranked local restaurants"*. Public alarms can be useful means of informing subscribers about hazardous road situations or heavy road congestion. According to the motion characteristics of the alarm target and alarm subscriber, we categorize spatial alarms into three classes: (1) moving subscriber with static target, (2) static subscriber with moving target, and (3) moving subscriber with moving target.

**Example 1: Location-based advertisements.** Macy's store in midtown Atlanta may set

a spatial alarm around its store location for *"sending e-coupons for a 20% discount to all gold members within a five mile radius of it store location"*. This allows the store to limit delivery of coupons to customers in the vicinity of the store. Customers may choose to subscribe to spatial alarms installed on Macy's store or otherwise, thus, personalizing delivery of information at their end. This is an example of a public alarm with moving subscribers and static target.

**Example 2: Location-based reminders [97].** A user may set a spatial alarm on her favorite grocery store for *"reminding her to buy some groceries when she is one mile away from the store over the weekend"*. This is an example of a private spatial alarm on static target with moving subscriber.

**Example 3: Location-enhanced social networking.** The concept of location when integrated into social networking applications like Facebook [13] allows the ability to add physical presence-based functionality. For example, a user may install an alarm on all friends which *"informs her whenever any of her friends are within a two mile vicinity of her current location during office lunch hours"*. As this requires monitoring of friends' location information, dealing with location privacy issues is also essential for such alarms [27]. This is an example of a private spatial alarm with moving targets and moving subscriber.

We develop a pub/sub system which allows a large number of mobile users to install a large number of different types of spatial alarms on a server. The server receives location data of all mobile users as well as other location sensitive data streams. This work focuses on various optimizations which can be performed to ensure system scalability and accuracy.

### 1.1.2 Location Privacy Constraints

Though LBSs provide many new opportunities, the ability to locate mobile users also presents new threats – the intrusion of location privacy [34, 53]. According to [34], location privacy is defined as the ability to prevent unauthorized parties from learning one's

current or past location. Location privacy threats refer to the risk that an adversary can obtain unauthorized access to raw location data, derived or computed location information by locating a transmitting device, hijacking the location transmission channel and identifying the subject using the device [54]. For example, location information can be used to spam users with unwanted advertisements or to learn about users' medical conditions, unpopular political or religious views. Inferences can be drawn from visits to clinics, doctor's offices, entertainment clubs or political events. Public location information can lead to physical harm, such as stalking or domestic abuse. The most important factor for the success or failure of any upcoming technology is its acceptance by the users. Any technology which makes people uncomfortable is very likely to be rejected or would at least progress at a slow rate. LBSs face this challenge in the form of privacy threats posed by them; this may prevent their widespread acceptance.

Users need to provide their personal information (i.e. location information) in order to avail the advantages provided by LBSs. The fact that pieces of their location information can be threaded together to infer daily routines is likely to make people uncomfortable and hesitant in using LBSs. Location privacy threats such as *location tracking*, *restricted space identification* as well as *observation identification* (as identified by [53]) need to be handled effectively to provide location privacy to users which would enable widespread adoption of the technology.

Several approaches have been proposed for protecting the location privacy of a user. We classify these techniques into three categories: (1) Location protection through user-defined or system-supplied privacy policies; (2) Location protection through anonymous usage of information; and (3) Location protection through pseudonymity of user identities, which uses an internal pseudonym rather than the user's actual identity. For those LBSs that require true user identity, strong security mechanisms such as location authentication and authorization have to be enforced in conjunction with their location privacy policy. In this work, we concentrate on the class of location-based applications that accept pseudonyms

and present the PRIVACYGRID framework for performing personalized anonymization of location information through customizable location $k$-anonymity and location $l$-diversity, thus enabling anonymous location-based queries in mobile information delivery systems.

Perfect privacy is clearly impossible as long as communication takes place. An important question here is *how much privacy protection is necessary?* Moreover, users often have varying privacy needs in different contexts. We propose to use quantitative metrics to model the location privacy requirements of a mobile user, firstly, using a random way point model and then taking the underlying road network into consideration.

## *1.2 Technical Challenges*

In this dissertation, we aim to handle the following technical challenges for scalability in location-based services:

### 1.2.1 High Scalability and High Accuracy for Spatial Alarm Processing Systems

Processing of spatial alarms requires meeting two demanding objectives: high accuracy, which ensures no alarms are missed, and high scalability, which guarantees that alarm processing is highly efficient and scales to large number of spatial alarms and growing base of mobile users. The conventional approach to similar problems involves periodic evaluations at a high frequency. Each spatial alarm evaluation can be conducted by testing whether the user is entering the spatial region of the alarm. High frequency is essential to ensure that none of the alarms are missed. Though periodic evaluation is simple, it can be extremely inefficient due to frequent alarm evaluation and the high rate of irrelevant evaluations. This is especially true when the mobile user is traveling in a location that is distant from all her location triggers, or when all her alarms are set on spatial regions that are far apart from one another.

### 1.2.2 Limiting Resource Consumption at Mobile Clients

We develop a server-centric approach and a distributed architecture for spatial alarm processing which is essential for extending the technology to clients using cheap location detection devices which may not possess significant computational power. Safe period and safe region optimizations are developed to scale the system and limit resource consumption at mobile clients. Even for clients with significant computing resources, energy and bandwidth consumption remain major bottlenecks and numerous works have dealt with the problem of energy conservation in mobile devices [44, 45, 86]. A main challenge in the design of our distributed architecture for spatial alarm processing is the need for developing *safe region* computation techniques that can provide a careful trade-off between server load and client energy consumption by taking into account: (i) the bandwidth required to communicate the safe region from the server to its corresponding mobile client, and (ii) the computation cost at a mobile client for monitoring its position with respect to the safe region.

### 1.2.3 Fast and Effective Techniques for Location Cloaking

We develop fast and effective cloaking algorithms for providing location $k$-anonymity and location $l$-diversity while maintaining the utility of LBSs under the PRIVACYGRID architecture. It is essential to develop fast, efficient cloaking algorithms which do not interfere with the completion of LBS requests. We propose to develop similar solutions for road network-based privacy protection.

### 1.2.4 Trade-offs between Privacy and Utility of LBSs

In PRIVACYGRID, we stress that location perturbation algorithms should be capable of dynamically making trade-offs between privacy and QoS. Unnecessarily large cloaking boxes will lead to not only poor QoS for the mobile users but also larger result sets to be transported from the corresponding LBS provider and higher processing costs for filtering

at either the mobile client side or at the location anonymizer, inevitably leading to larger delays for obtaining useful query results. We also propose to develop solutions which are able to balance privacy-utility trade-offs in the presence of the underlying road network.

## *1.3 Dissertation Scope and Contributions*

This dissertation makes the following contributions in order to address the challenges described in the previous section.

**High Scalability and High Accuracy for Spatial Alarm Processing Systems:** Any server-based approach must allow optimizations for processing spatial alarms installed by multiple mobile clients in order to ensure system scalability. We make the following contributions towards developing scalable server-based solutions for spatial alarm processing.

*Safe Period-based Processing:* We optimize the conventional approach of periodic alarm processing by advocating a motion-aware *safe period-based* alarm evaluation framework. Concretely, we formalize the concept of spatial alarms and the problem of spatial alarm processing. We introduce the concept of *safe period* to minimize the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system. We show that our safe period-based alarm evaluation techniques can significantly reduce the server load for spatial alarm processing compared to the periodic evaluation approach, while preserving the accuracy and timeliness of spatial alarms. Furthermore, we develop alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, aiming at optimizing safe period computation at the server. We evaluate the scalability and accuracy of our approach using a road network simulator and show that our proposed framework offers significant performance enhancements for the alarm processing server while maintaining high accuracy of spatial alarms compared to the conventional periodic alarm evaluation approach [28].

*Safe Region-based Processing:* We propose a distributed architecture and a suite of safe region techniques for scalable processing of spatial alarms [26]. We show that safe

region-based processing enables resource optimal distribution of partial alarm processing tasks from the server to the mobile clients. “Resource optimal” implies that our distributed architecture minimizes unnecessary alarm evaluations at both server and mobile clients. We propose three different safe region computation algorithms to explore the impact of size and shape of the safe region on network bandwidth, server load and client energy consumption.

**Limiting Resource Consumption at Mobile Clients:** Our suite of safe region computation techniques allows us to analyze the impact of the size and shape of safe region on the client-server communication cost, server load and client energy consumption. Concretely, we develop three safe region computation techniques: (i) *Maximum Weighted Perimeter Rectangular Safe Region*, (ii) *Grid Bitmap Encoded Safe Region*, and (iii) *Pyramid Bitmap Encoded Safe Region*. These alternative methods for safe region computation provide flexible support for mobile clients with heterogeneous capabilities in terms of CPU, network bandwidth and energy capacity.

**Fast and Effective Techniques for Location Cloaking:** We present a three-tier architecture to provide location privacy using a third party anonymizer service. We develop location cloaking algorithms which are fast and capable of keeping the perceived delays due to location anonymization to a minimum. In order to achieve this goal, we maintain a simple grid-based data structure which keeps the mobile object counts in each cell. We develop *dynamic* bottom-up and top-down grid cloaking algorithms with the goal of achieving high anonymization success rate and efficiency in terms of both time complexity and maintenance cost. A hybrid approach that carefully combines the strengths of both bottom-up and top-down cloaking approaches to further reduce the average anonymization time is also developed.

**Trade-offs between Privacy and Utility of LBSs:** We allow each user to specify her privacy requirements in terms of privacy and utility constraints. In PRIVACYGRID, we use

location $k$-anonymity and location $l$-diversity as two quantitative metrics to model the location privacy requirements of a mobile user. In the context of LBSs and mobile users, location $k$-anonymity refers to $k$-anonymous usage of location information. A user is considered location $k$-anonymous if and only if the location information sent from the mobile user to a LBS is indistinguishable from the location information of at least $k - 1$ other users. Location $l$-diversity is introduced to strengthen the privacy protection of location $k$-anonymity in situations where location information shared by the $k$ users is sensitive. By increasing $l$ value to two or higher, it significantly reduces the probability of linking a static location or a symbolic address (such as church, restaurant, doctor's office) to a mobile user. As QoS measures, we use *maximum spatial resolution* which allows the mobile user to control the spatial resolution reduction within an acceptable QoS specific range. It can be changed or adjusted according to the type of location service, the time of day, month or year, and on a per message level. Similarly, the fourth measure is *maximum temporal resolution*, which controls the temporal delay acceptable for maintaining the desired QoS. Our algorithms use these constraints effectively to balance the privacy utility trade-offs.

## *1.4 Organization of the Dissertation*

This dissertation is organized as a series of chapters each dealing with one of the different problems described above. Each chapter details the core problem being addressed, provides basic concepts and then describes the development of a solution followed by the evaluation of the proposed solution. Relevant work is described along with each chapter. Concretely, the dissertation is organized as follows.

In chapter 2, we present a *location-centric framework* that advocates a clean separation of static location data from moving objects in terms of both indexing structure and location query processing. Employing a separate index for moving objects enables us to minimize the index maintenance cost in the presence of location updates of moving objects,

and speeds up the evaluation of location queries by maximizing the amount of parallel processing, specifically processing location queries over static objects via location index and processing location queries over moving objects through grid index. We evaluate the performance of a system processing different types of location queries by maintaining a location index for indexing of static objects and a grid-based framework for updating and indexing positions of moving objects. Our experimental results demonstrate the advantages of our location-centric framework for processing location queries and conform with the predicted results obtained via analysis in terms of IO costs for location index and object index.

In chapter 3, we present a motion-aware safe period framework and a suite of optimization techniques for scalable processing of spatial alarms. The chapter makes two important contributions towards supporting spatial alarm-based mobile applications. First, we introduce the concept of *safe period* to minimize the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system. We show that our safe period-based alarm evaluation techniques can significantly reduce the server load for spatial alarm processing compared to the periodic evaluation approach, while preserving the accuracy and timeliness of spatial alarms. Second, we develop a suite of spatial alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, which reduces the safe period computation cost for spatial alarm evaluation at the server side. We evaluate the scalability and accuracy of our approach using a road network simulator and show that the proposed motion-aware safe period-based approach to spatial alarm processing offers significant performance enhancements for alarm processing on server side while maintaining high accuracy of spatial alarms.

In chapter 4, we make three important contributions towards supporting efficient processing of spatial alarms. First, we introduce the concept of safe region-based alarm processing to enhance the scalability of the system. We develop different techniques for safe region computation, namely the maximum weighted perimeter rectangular safe region approach and the two BSR approaches, GBSR and PBSR. Second, we consider trade-offs

between various heuristics for safe region computation based on size, shape of the region which affects the client to server communication cost, alarm evaluation cost, downstream bandwidth consumption and the client energy consumption. Our experimental evaluation shows that safe region techniques outperform other alarm processing techniques like periodic evaluation, safe period-based approach and offer close to optimal performance for different alarm distribution scenarios. Last but not the least, our framework supports heterogeneous environments with varying server load, resource conditions and heterogeneity of client capabilities using the concept of bitmap encoded safe regions.

In chapter 5, we present the SLIM system as an efficient solution for the mobile information monitoring problem in presence of non-spatial attributes. This chapter makes three important contributions towards efficiently solving the information monitoring problem in presence of spatial as well as non-spatial attributes. First, we show that the addition of less dynamic non-spatial attributes to the mobile information mix, although it leads to a larger amount of data updates being handled at the processing server, provides opportunities to enhance system scalability beyond what is possible with spatial attributes alone. We propose the concept of selective evaluation of received data updates which allows us to determine data updates that may be dropped without further processing. Second, we propose the concept of safe containment which allows us to perform the selective processing of data updates. The selective processing approach seeks cooperation from mobile users and participating data sources in the information monitoring process but provides heavy returns in terms of savings in communication costs. We also present efficient algorithms for safe value container computation for single-dimensional and multi-dimensional data. Last but not the least, we conduct extensive experimental evaluation for a real world road network-based simulator which shows that safe containment allows for savings in terms of energy and bandwidth consumption for mobile clients in a wireless environment. Computation costs at the server are also considerably reduced, thus enhancing system scalability.

The next two chapters are focused on developing solutions for enabling privacy-aware

delivery of location-based services. In chapter 6, we describe the PRIVACYGRID framework which allows users to express their privacy requirements in terms of location hiding and QoS measures to control query processing overheads. Three dynamic grid-based spatial cloaking algorithms are developed for providing location $k$-anonymity and location $l$-diversity in a mobile environment. A brief discussion of the PRIVACYGRID mechanisms for processing anonymous location queries is provided. We report our extensive experimental evaluation results and show that compared to existing grid cloaking approaches such as [82], our dynamic grid cloaking algorithms provide much higher anonymization success rate and yet are highly efficient in terms of both time complexity and update cost.

In chapter 7, we present MOBICLOAK, a road network-aware location anonymization model for protecting location privacy under road network mobility models. The MOBICLOAK design exhibits three unique features. First, we enrich user $k$-anonymity by introducing segment $s$-anonymity as a companion metric for guarding location privacy of mobile users on road networks. Second, we promote the use of graph density as an important measure for determining optimal cloaking subgraphs in a road network. In addition, we devise a suite of graph-based cloaking algorithms, supporting various levels of anonymization optimization in terms of trade-offs between privacy and utility of cloaked location. Our experimental evaluation demonstrates that our mobility-aware expansion combined with controlled randomization provides highly efficient location anonymization with high attack resilience compared to other existing approaches.

Chapter 8 concludes this dissertation with a summary of unique contributions of this dissertation research and a discussion of open issues and possible research directions facilitated by this work.

# CHAPTER II

# LOCATION-CENTRIC DATA MODELING FOR SCALABLE LOCATION-BASED SERVICES

## *2.1 Introduction*

The world of computing is changing rapidly. Wireless connectivity and mobility are no longer considered a luxury but are a necessity in our everyday life. With rapid advances in ubiquitous connectivity people, vehicles and computers are connected at all times and location-aware computing will become a critical capability in future computing environments. Monitoring and evaluating location queries over static and moving objects is a fundamental functionality for many mobile location-based applications, ranging from location-based emergency response, fleet management, cargo tracking, child care, to location-based advertisement and location-based entertainment. Such systems typically consist of a distributed collection of database servers, base stations, application servers and a large number of static and mobile objects. The database server manages the location data of mobile users and static objects, such as gas stations, restaurants, etc. Location queries are registered and evaluated at the database server. In these location-based computing systems two predominant costs determine the system performance and scalability: (1) the wireless communication cost for location updates of mobile clients and (2) the location query evaluation cost at the database server.

Location queries can be classified into three major categories depending on the motion properties associated with the querying object or the target objects of a query. The first class of queries consists of *moving queries over static objects*. A query such as "Provide locations of all gas stations within 10 miles, selling gas at less than three dollars per gallon, over the next hour" belongs to this particular class of queries. The *focal object* of the query

is the car moving on the highway which poses this query. On the other hand a query such as "Provide the locations for all cabs within five miles of the downtown area of city A" is a *static query over moving objects*. A query such as "Locate all customers who are looking for cabs and are within five miles of my position over the next 20 minutes" is an example of a *moving query over moving objects*. The cab which registers this query is the *focal object* of the query in this scenario.

Most existing research efforts to date have been dedicated to spatial and temporal methods for indexing objects or indexing queries [92, 94, 68, 50]. The goal of these indexing techniques is to provide fast processing of location queries in the presence of moving objects, while ensuring the correctness of the query results within certain tolerance bounds. Even though some indexing techniques such as TPR tree [94] have shown higher effectiveness in indexing moving objects than static objects, few studies provide an in-depth understanding of the potential performance benefit of separating the indexing over static objects from the indexing for moving objects. Furthermore, most of the existing object indexing schemes have to deal with the increased time complexity as the number of mobile objects becomes large or the velocity of the objects varies significantly in the system.

We argue that a clean separation of query processing over static objects from query processing over moving objects not only helps reduce the complexity of the problem, but also provides significant performance gains for scaling the processing of queries over a mixed workload of both types of objects. Processing requirements for evaluating location queries over static objects are significantly different from those for queries over moving objects. For example, queries over static objects are typically issued by mobile clients on the move. The target objects of the queries are static objects, but the spatial query range changes as the query issuer (i.e., the focal object of the query) moves on the road. Thus, the set of static objects to be retrieved from the database is highly dependent on the motion behavior and the current location of the query focal object. In contrast, queries over moving objects do not involve any static objects, but need to continuously track the positions of

moving objects in the vicinity of the focal object. The ability to efficiently track the precise positions of the target moving objects is critical for processing queries over moving objects in terms of both result quality and system performance.

Bearing these observations in mind, we develop a location indexing framework to advocate a clean separation of static objects from moving objects in terms of both spatial indexing structure and location query processing. By promoting locations as *first class citizens*, we show that the location indexing framework enables scaling of location-based services. Concretely, we propose to build a location index for managing all the static objects in terms of their geographical locations in the real world. Our formal analysis and experimental evaluation both show that our location indexing framework can provide fast access capability with low maintenance cost compared to the existing object indexing schemes. There are several factors for such performance gains. First, location index is served as a primary clustered index, in the sense that all static objects stored on disk are ordered by their locations instead of their object identifiers, thus requiring fewer disk IOs compared to the existing object indexing schemes. Second, by managing moving objects, their location updates, and queries over moving objects using a separate object index, we considerably reduce the retrieval and maintenance costs of indexing structures for both static and moving objects.

Concretely, the processing efficiency of location queries over static objects can be greatly enhanced along several dimensions, including reduced index size, improved disk locality, and fast searches at varying granularity of spatial approximations. Also, by employing a grid-based indexing scheme for processing location queries over moving objects, we can reduce index maintenance cost in the presence of location updates of moving objects and speed up query evaluation by maximizing the amount of parallel processing, specifically using a location index for queries over static objects and grid index for those over moving objects.

We conduct a formal IO cost analysis and an in-depth experimental comparison of our

location indexing framework with the conventional object indexing approaches and show that the location-indexing approach prevails over existing object indexing schemes under all situations. In some scenarios the use of location index requires as low as only $6\%$ of the IOs required for query evaluation by a corresponding object index.

The rest of the chapter is organized as follows. We begin with a motivating example (Section 2.2) showing the problems with object indexing that mixes static objects and moving objects. Next, we introduce the basic concepts associated with our system in Section 2.3. We then discuss the advantages of our location-centric framework. We describe our approach for building a location-centric framework, including a formal analysis of the IO cost model for our location indexing scheme in Section 2.4. The grid-based model for processing location queries over moving objects is presented in Section 2.5. Section 2.6 reports the experimental evaluation of our location-centric indexing approach and compares it with existing object indexing schemes. A discussion of related work is presented in Section 2.7.

## *2.2 Motivation and Problem Statement*

In this section, we lay the groundwork for motivating our location-centric framework for location-dependent data. An example is used to illustrate the key differences in terms of processing requirements between our location centric framework and the object-centric approach, and the potential inefficiencies associated with the object-centric framework, especially in the context of processing location queries over static objects.

Consider the example as shown in Figure 1, which displays a set of static objects in the *Universe of Discourse U* (or map). Each object in *U* has a corresponding object entry in the object table *O*. In an object-centric framework, the data maintained in the object table is indexed by the *'object ID'* using a R-tree based object index which uses *minimum bounding rectangles* (MBRs) to index the objects. Consider a location based query *q1* over this object table as shown in Figure 1. Assuming that an R-Tree based object index exists,

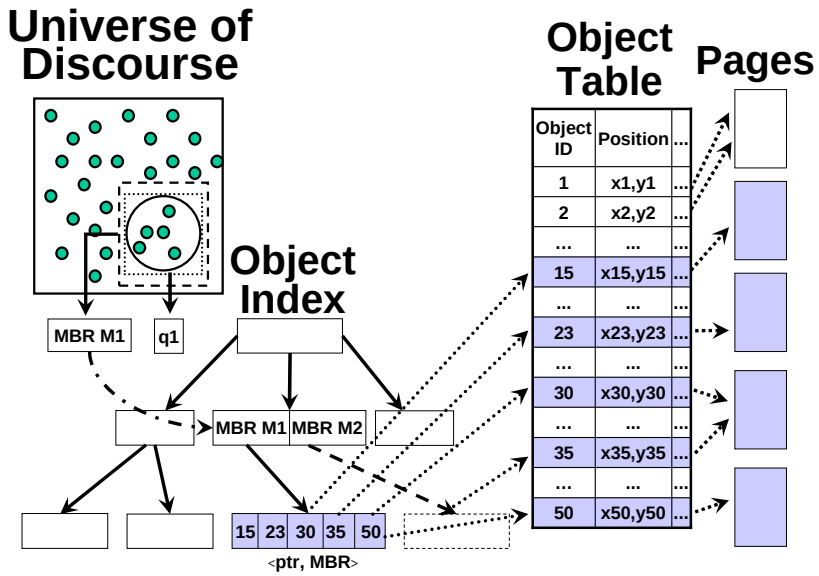



**Figure 1:** Object-based Modeling

the query retrieves a set of index entries that correspond to the list of objects belonging to the region being queried by *q1*. Now for further query processing the object entries need to be retrieved from the table using this index on the object ID column. However, since the data organization on disk is based on object ID rather than spatial locality, the objects being queried, though in close vicinity, may be spread across multiple disk pages resulting in highly inefficient disk bandwidth utilization. For instance in this example, five objects reside in the region being queried by *q1*. Thus as many as four different *disk pages* may need to be fetched. Furthermore, if we incorporate moving objects into this model some of the MBRs may need to be split into smaller ones as new mobile objects join the system, not only increasing the number of MBRs to be maintained in the R-Tree based object index and the index search cost, but also increasing the maintenance cost of the index since the location updates of moving objects often cause the mobile objects to be moved from one MBR to another.

We reach the following conclusions from the above discussion. First, when the system needs to maintain an index for large number of static and moving objects, such maintenance cost can be significant, since each position update of a moving object requires at least one

index search and one update at each of the corresponding index nodes. Second, the spatial locality of the objects is the dominating property for both static and moving objects in a mobile environment. Location queries typically attempt to retrieve objects of interest within a particular spatial region in the vicinity of the focal object. Thus static objects or moving objects within certain spatial vicinity are typically accessed together. Access of static objects is often separated from the access of moving objects due to the fact that location queries are typically targeted at either static objects or moving objects, but rarely targeted at both types. Thus mixing static and moving objects in one index is not an optimal solution for processing location queries. Last but not the least, object-based indexing and storage of static objects fail to capture the spatial locality-based access patterns for both disk access and processing of queries over static objects. Thus, object indexing and object-based disk management will lead to inefficient retrieval of large number of irrelevant object tuples in order to find the target objects relevant to a location query.

In order to benefit from the properties associated with static objects, we develop a location indexing framework that cleanly separates static objects from moving objects. The basic principles for making such a clean separation include (1) creating and maintaining separate indexing structure for static objects and moving objects; (2) developing location index to speed up the retrieval of static objects where location is used as the *primary key* instead of objects for both index search and disk access; and (3) developing a dedicated object indexing structure for managing moving objects.

Our location indexing framework offers several advantages. First, it offers significant performance gains by separating operations over moving objects from the retrieval of static objects. Second, the location indexing framework encourages higher level of parallel processing since queries over static objects can be processed independently from queries over moving objects, leading to another level of performance and throughput enhancement. Figure 2 displays our approach for location-centric modeling of static objects. Our approach first divides the entire universe of discourse into *spatial regions* based on factors such as

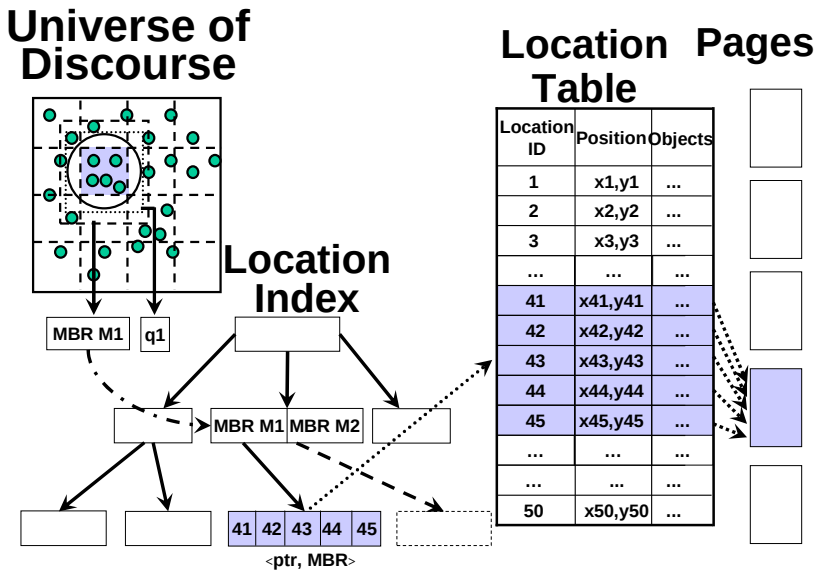



**Figure 2:** Location-centric Modeling

spatial locality, object density and page size. The example in Figure 2 displays a simple grid-based division of the region. The static object data is organized in a location-centric manner with spatial locality of objects being mapped to locality of the data on the disk. By enabling the use of spatial locality to access static object data on disk, we improve query processing by making more efficient use of disk bandwidth. For example, as shown in Figure 2, the data for the five objects relevant to query *q1* can now be retrieved in a single page fetch. Armed with these insights for location-dependent data, we next describe our framework for modeling and indexing static data. But first, we describe the basic concepts associated with our location-centric framework.

## *2.3 Basic Concepts*

The basic elements of our system model are a set of moving or stationary objects and a set of moving or static (range) queries. Fast evaluation is critical for processing location queries, as it not only improves the freshness of the query results by enabling more frequent reevaluation, but also increases the scalability of the system by enabling timely evaluation of a large number of queries over a large number of objects.

We denote the set of moving or stationary objects as $O$, where $O = O_m \cup O_s$ and $O_m \cap O_s = \emptyset$, $O_m$ denotes the set of moving objects and $O_s$ denotes the set of stationary objects. We denote the set of moving or static queries as $Q$, where $Q = Q_m \cup Q_s$ and $Q_m \cap Q_s = \emptyset$, $Q_m$ denotes the set of moving location range queries and $Q_s$ denotes the set of static location range queries. Some essential concepts associated with our location-based framework are defined below.

**Universe of Discourse (UoD):** We refer to the geographical area of interest as the universe of discourse (or map), which is defined by $U = Rect(x, y, w, h)$, where $x$ is the x-coordinate and $y$ is the y-coordinate of the lower left corner of a rectangular region, $w$ is the width and $h$ is the height of the universe of discourse. Basically, we consider maps which are rectangular in shape. In case our Universe of Discourse is extremely large, it is possible to divide the entire Universe of Discourse(UoD) into smaller maps (or UoDs). Each UoD can be handled by a location-based system exclusively setup for managing this particular Universe of Discourse.

**Static Objects:** Any static object $o_s \in O_s$ is represented by a tuple: $\langle i_o, \vec{p}, att_p \rangle$ in the object table $O$. Here, $i_o$ is the unique object identifier, $\vec{p} = (p_x, p_y)$ identifies position of the static object in the UoD where $p_x$ is its position in the $x$-dimension and $p_y$ is its position in the $y$-dimension, and $att_p$ denotes a set of attributes related to the object. The position $\vec{p}$ of the object may also be represented by a *semantic location* such as a postal address rather than as two-dimensional coordinates in the UoD. The set of properties $att_p$ may represent multiple attributes like associated businesses (e.g. restaurant, hardware store, gas station etc.).

**Locations:** We introduce the concept of location tuples where any location can be represented by $\langle i_l, \vec{p}, \vec{oid}, \vec{att}_p \rangle$ in the location table $L$. Here, $i_l$ is the unique location identifier assigned to this location tuple, $\vec{p} = (p_x, p_y)$ identifies *position of the location* in the UoD where $p_x$ is the x-coordinate of the location and $p_y$ is the y-coordinate for the location, $\vec{oid}$

is a vector (of object identifiers) denoting the set of static objects associated with this location and $\vec{att}_p$ is a vector of attributes denoting a set of properties about the corresponding objects. As in the case of static objects, the position $\vec{p}$ describing the location may also be represented by a *semantic location* such as a postal address rather than as two-dimensional coordinates in the UoD. Locations at the user interface level can be *semantic locations* or *geometric locations* but at the conceptual level they are generalized to the geo-spatial model for processing by the system. The vector associated with the set of properties $\vec{att}_p$ may represent multiple attributes like associated businesses (e.g. restaurant, shopping complex, gas station etc.).

**Moving Objects:** We describe a moving object $o_m \in O_m$ using a tuple: $\langle i_o, \vec{p}, \vec{v}, att_p \rangle$ in the object table $O$. Here, $i_o$ is the unique object identifier, $\vec{p} = (p_x, p_y)$ is the current position of the moving object where $p_x$ is its position in the $x$-dimension and $p_y$ is its position in the $y$-dimension, $\vec{v} = (v_x, v_y)$ is the current velocity vector of the object, and $att_p$ is a set of properties about the object.

**Static Queries:** We describe a static query $q_s \in Q_s$ using a tuple: $\langle i_q, region, f \rangle$. Here, $i_q$ is the unique query identifier, $region$ is the rectangular or circular region defining the shape and extent of the spatial query region, and $f$ is a predicate, called *filter*, defined over the properties ($att_p$) of the target objects of the query. Further, the query specifies whether it is searching for static objects or moving objects which can also be easily determined from the filter for the query. For example, a query related to restaurants is obviously looking for static objects and similarly a query related to cabs is looking for moving objects. In real life scenarios, it is difficult to find queries which search for both static and moving objects; most queries are directed either over static objects or over moving objects. This further validates our decision to separate processing of queries over static and moving objects.

**Moving Queries:** We describe a moving query $q_m \in Q_m$ by a quadruple: $\langle i_q, i_o, r, f \rangle$. Here, $i_q$ is the unique query identifier, $i_o$ is the object identifier of the focal object of the query, $r$ defines the shape of the spatial query region bound to the focal object of the query,

and $f$ is a predicate, called *filter*, defined over the properties ($att_p$) of the target objects of the query. Note that, $r$ can be described by a closed shape description such as a rectangle or a circle. This closed shape description also specifies a binding point, through which it is bound to the focal object of the query. In the rest of the chapter we assume that a moving location query specifies a circle as its range with its center serving as the binding point and we use $r$ to denote the radius of the circle. Similar to the case of static queries, any moving query $q_m$ is defined either over the set of static objects $O_s$ or over the set of moving objects $O_m$.

We further define a grid-based framework on top of the existing universe of discourse to handle position updates for moving objects. Some relevant concepts related to this grid based framework are defined below.

**Grid and Grid cells:** In our framework, we map the universe of discourse $U = Rect(x, y, w, h)$ onto a grid $G$ of cells. Each grid cell is an $\alpha \times \alpha$ square area, where $\alpha$ is a system parameter that defines the cell size of the grid $G$. Formally, a grid corresponding to the universe of discourse $U$ can be defined as $G(U, \alpha) = \{A_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N, A_{i,j} = Rect(x+i*\alpha, y+j*\alpha, \alpha, \alpha), M = \lceil w/\alpha \rceil, N = \lceil h/\alpha \rceil\}$. $A_{i,j}$ is an $\alpha \times \alpha$ square area representing the grid cell that is located on the $i$th column and $j$th row of the grid $G$.

**Position to Grid Cell Mapping:** Let $\vec{p} = (p_x, p_y)$ be the position of a moving object in the universe of discourse $U = Rect(x, y, w, h)$. Let $A_{i,j}$ denote a cell in the grid $G(U, \alpha)$. $Pmap(\vec{p})$ is a position to grid cell mapping, defined as $Pmap(\vec{p}) = A_{\lceil \frac{p_x - x}{\alpha} \rceil, \lceil \frac{p_y - y}{\alpha} \rceil}$.

**Current Grid Cell of a Moving Object:** Current grid cell of a moving object is the grid cell which contains the current position of the moving object. If $o_m \in O_m$ is an object whose current position, denoted as $\vec{p}$, is in the Universe of Discourse $U$, then the current grid cell of the object is formally defined by $curr\_cell(o_m) = Pmap(\vec{p})$.

## *2.4 Location-Centric Framework*

In this section we describe our approach towards modeling and indexing location-dependent data in terms of location tuples rather than object tuples. Our approach modifies the object-based modeling and indexing approach to emphasize on modeling static objects with their location as the primary key instead of object identifiers. As a result, each relevant location in our model has one or more static objects associated with it. A location-centric model for static objects requires: (1) modeling object tuples as location tuples and, (2) constructing a location index to efficiently answer queries associated with static objects.

### 2.4.1 Location Centric Data Modeling

In the location-centric framework, the first step for indexing static objects is to model them differently with location as the *primary key* instead of objects. This remodeling involves grouping location data in a manner which allows spatial locality of data to be mapped to locality of data on disk.

Our location-centric framework stores and retrieves static objects in terms of their spatial locations. In order to uniquely identify the static objects using their spatial locations, we build a grid-based overlay on top of the geographical area of interest, namely the Universe of Discourse, and use this grid-based layout to determine the static objects that can be stored and retrieved together.

The task of creating a location table consists of the following steps.

**Step 1 - Determining grid partition parameters:** The first step in building a location table is to determine the grid partition parameter $\beta$, which determines the size of each cell depending on the maximum number of locations permitted in a single cell. In general, the grid may consist of cells of different sizes dependent upon the distribution of locations of interest in the Universe of Discourse. Areas in the Universe of Discourse, which have higher density of relevant locations, will need to be divided into smaller cells to ensure that the number of locations in each cell does not exceed the system-defined maximum

limit. The decision on the cell size is based on a number of factors, including the density distribution of the static objects, the page size of the disk access, the size of a location tuple, to name a few. The goal is to accommodate the data related to all location points within one cell in a single disk page.

**Step 2 - Obtaining relevant locations:** Location can refer to a position or a spatial region. In our framework, each location entry in the location table refers to a spatial region represented in terms of grid cell. Each entry in the location table is uniquely identified by its location ID and may refer to multiple static objects. We scan the static object table and obtain relevant locations in terms of the grid cell in which they reside. Alternatively, we may have a set of relevant locations available elsewhere, which we want to insert into the location table. For example, we may want to insert relevant locations from a yellow book or as identified by some other location service.

**Step 3 - Generating location identifiers:** In the next step, we map each location entry to a single cell on the grid and assign a *location identifier (lid)* to uniquely identify the list of locations within each cell. Each location in a single cell is assigned a unique *lid*, with all locations belonging to a particular cell being assigned location identifiers in sequential order. The order of the location identifiers will also determine the order in which the tuples are arranged on disk. Not that ordering of the tuples is important for efficient retrieval of static object data as discussed in section 2.2.

**Step 4 - Associating objects with each location:** In the final step, all static objects associated with a location entry in the location table are assigned to the location. We store the object identifier $oid$ and associated values for other attributes of the static objects as vectors related to this particular location tuple.

In case of skewed distributions, we can construct a quadtree-based structure instead of grid structure. Also, if a particular location tuple has a large number of associated objects leading to an overflow, we can create an overflow table and provide a pointer to this overflow table which will store the list of objects associated with this location.

### 2.4.2 Location Index

In this section we give a detailed description of the various aspects of the location index. A formal analysis of the IO costs associated with the location index is provided in the next section.

The design of our location indexing scheme follows a number of basic principles. First, we model spatial locations in terms of two dimensional geographical coordinates and extend the R-tree data structure [55] to model locations as first class citizens. For example, leaf nodes in our R-tree based location index refer to a set of locations instead of a set of objects. Second, we build the R-tree based location index bottom up through sorting rectangles and merging nearby rectangles to form a hierarchical indexing tree structure. There are several ways to sort the MBR rectangles, such as tree-based variations [55, 96, 31, 32, 95] and methods that use space filling curves [63, 59]. According to [63], two dimensional Hilbert curve through centers only (2D-c Hilbert) achieves the best clustering among all space filling curve algorithms. In this algorithm, each data rectangle is represented by its center only. The Hilbert value of the center is the Hilbert value of the rectangle. The third design principle for building a high performance location index is to balance the access latency to all locations. We build a location index in three steps. Figure 3 illustrates the building of a location index by example.

**Step 1 - Determining MBRs associated with each location:** The first step in building a location index for fast retrieval of static objects is to determine the MBR associated with each location identifier. Figure 3 displays a set of locations *(L1, L2,..., L9)*, where each location is represented by its MBR. A location can be a region of any shape and be approximated by its *MBR*.

**Step 2 - Constructing R-tree:** By sorting the *MBRs* for locations, an R-tree is constructed bottom up for indexing the unique locations. The leaf nodes in the R-tree structure contain entries of the form $\langle ptr, MBR \rangle$, where $ptr$ is a pointer referring to a particular location entry and $MBR$ is the minimum bounding rectangle enclosing this location. Intermediate

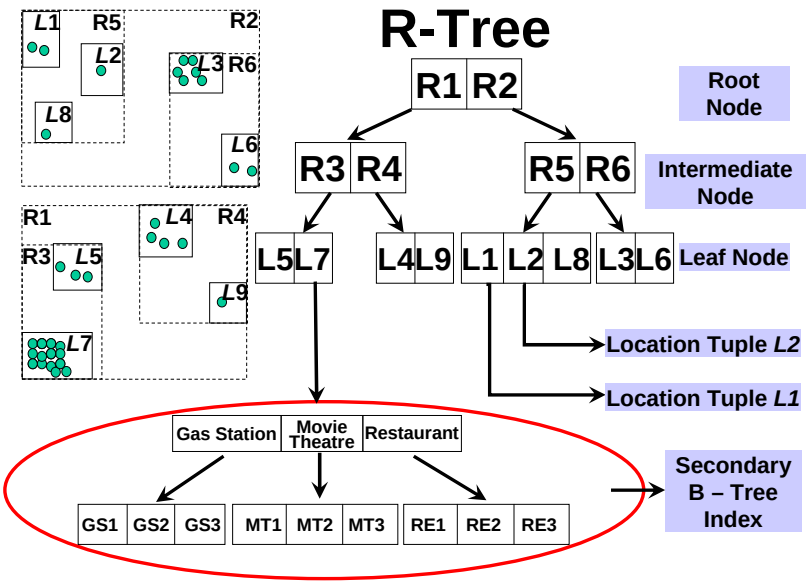



**Figure 3:** Location Index – An Example

level nodes contain entries of the form $\langle childptr, MBR \rangle$, where $childptr$ is a pointer to a lower level node in the R-tree and $MBR$ is a region enclosing the MBRs for all entries in the child node.

**Step 3 - Adjusting leaf node pointers:** Some leaf nodes may have locations associated with a large number of static objects (hundreds or thousands of offices, stores, restaurants, for example). One way to handle this situation is to create a secondary index over some other attribute associated with the static objects at this location, speeding up the access to these objects. The attribute that is frequently used in the filter conditions of location queries is a natural candidate to build such a secondary index. For example, in Figure 3 location *L7* has a large number of buildings associated with it. In this step, a second level B-tree index is constructed over the attribute *Associated Business* for the static objects associated to the *L7* location, allowing us to access these objects in alphabetical order of their *Associated Business*. The pointers for leaf nodes referring to such locations are directed at the root node of the secondary B-tree index over the static objects. For other locations in the leaf nodes like *L1* or *L2*, which have only a few objects associated with each location, the leaf node points to the location tuple that contains these objects. In general secondary B-tree

indices are maintained for locations with large number of static objects. The decision is primarily based on the trade-offs involved in maintaining B-tree indices and the advantage gained in query performance. Our current simulator constructs B-tree indices for those locations that are associated with more than a specified number of objects.

An update on the location index requires a search on the R-tree index to first locate the relevant location which needs to be updated. The list of objects or the B-tree index on the objects is then appropriately updated. Note that the R-tree index is fixed and *ideally* no locations are added to or removed from the index.

### 2.4.3 Analytical Model for IO Cost Estimation

To better understand the performance gain of location index over object index, we provide an analytical model for estimating the performance difference between the location index and a corresponding object index. This analytical estimate is based on well studied R-tree access cost models [105] and our observation of the difference between the location-centric data model and object-centric data model for both storage and retrieval. Our comparison of location index and object index concentrates on the performance differences between these indices from three different perspectives: (1) the number of data rectangles indexed which determines the index size, (2) the number of node accesses required which determines the index search time, and (3) the disk access costs for accessing relevant relational data tuples for each index.

Table 1 provides a list of symbols which we use throughout the rest of this section. Note that we are considering the top-level R-tree structure for the location index here.

We consider a two-dimensional unit workspace $WS = [0, 1)^2$ which has $N_L$ data rectangles stored in the corresponding location index $L$. A selection query seeks all regions which intersect the query $q = (q_1, q_2)$ where $q_1$ and $q_2$ denote the width and height of the query window.

The density $D$ of a set of $N$ rectangles signifies the average number of rectangles that

**Table 1:** List of Symbols

| Symbol | Definition |
|---|---|
| $h$ | height of R-tree structure of location index $L$ |
| $f$ | node fan-out for top-level R-tree structure of $L$ |
| $N_L$ | number of data rectangles indexed in $L$ |
| $D_L$ | density of data rectangles indexed in $L$ |
| $N_L^l$ | number of nodes of $L$ at level $l$ |
| $D_L^l$ | density of node rectangles of $L$ at level $l$ |
| $N_{static}$ | average number of static objects per location in the dataset |
| $f_{oc}$ | average fraction of data rectangles in a leaf node of an object index which satisfy spatial constraints for a query $q$ |

contain a given point in the two-dimensional space. For the unit workspace $WS$, where each rectangle has an average extent of $s = (s_1, s_2)$, this density is defined as, $D(N, s) = \sum_N s_1 s_2 = N \cdot (s_1 \cdot s_2)$ [105].

We consider a location index $L$ of height $h$ where the root is at level $h$ and leaf nodes are assumed to be at level 1. The number of nodes at level $l$ are denoted by $N_L^l$ with average size $s_L^l$. In this case, the expected number of node accesses in order to answer a selection query using the query $q$ with window $(q_1, q_2)$ is given by

$$\lambda_{total}(L, q) = \sum_{l=1}^{h-1} intx(N_L^l, s_L^l, q) \tag{1}$$

where $intx(N_L^l, s_L^l, q)$ returns the number of nodes at level $l$ intersected by query window of $q$. The expected number of node accesses is equal to the expected number of intersected nodes at each level. This does not consider any access to the root node for the R-tree structure of the location index as the root node is expected to be stored in main memory. Now, given a set of $N$ rectangles $r_1, r_2, ..., r_N$ with average extent $s$ and a rectangle $r$ with extent $q$, the average number of rectangles $r_i$ intersected by $r$ is, $intx(N, s, q) = N \cdot (s_1 + q_1) \cdot (s_2 + q_2)$, which is equal to the number of a second set of $N$ rectangles with average extent $s' = (s_1 + q_1, s_2 + q_2)$ that contain a point in the workspace. This, by definition, equals the density of the second set of rectangles. Hence, $intx(N, s, q) = intx(N, s', 0) = D(N, s') = N \cdot (s'_1 \cdot s'_2)$, where $s'_1 = s_1 + q_1$ and

$s'_2 = s_2 + q_2$. Thus, we have

$$\lambda_{total}(L, q) = \sum_{l=1}^{h-1} N_L^l \cdot (s_{L,1}^l + q_1) \cdot (s_{L,2}^l + q_2). \quad (2)$$

To express (2) as a function of number of rectangles $N_L$ and density $D_L$ of the rectangles, instead of location index properties, further refinement is required. The height $h$ of the top-level R-tree structure for location index $L$ with average fan-out $f$ that stores $N_L$ rectangles is given by $h = 1 + \lceil log_f \frac{N_L}{f} \rceil$ [43]. Since a node contains $f$ rectangles on average, the average number of leaf nodes can be approximated as $N_L^1 = N_L/f$ and the average number of nodes at level $l$ is $N_L^l = N_L/f^l$. Using the *squaredness assumption* for the R-tree structure of location index $L$ [105], we have,

$$s_{L,k}^l = (D_L^l \cdot \frac{f^l}{N_L})^{\frac{1}{2}}, \quad (3)$$

$\forall\ k$. For the two-dimensional case we have $k \in 1, 2$. Further, the density $D_L^{l+1}$ of node rectangles at level $l + 1$ is a function of density $D_L^l$ of node rectangles at level $l$ [105]:

$$D_L^{l+1} = (1 + \frac{{D_L^l}^{\frac{1}{2}} - 1}{f^{\frac{1}{2}}})^{\frac{1}{2}}. \quad (4)$$

This enables us to calculate the density of rectangles at level $l$ in terms of the density of data rectangles $D_L$. By combining these equations the expected number of node accesses for a selection query can be calculated in terms of data set properties $N_L$ and $D_L$, fan-out $f$ of the R-tree structure of location index $L$ and the query window $q$ as,

$$\lambda_{total}(L, q) = \sum_{l=1}^{h-1} (D_L^l + \frac{q_1 \cdot q_2 \cdot N_L}{f^l} + (q_1 + q_2) \cdot (D_L^l \cdot \frac{N_L}{f^l})^{\frac{1}{2}})$$

Let $N_{static}$ denote the average number of static objects per location in the data set. This implies that if a dataset has $N_L$ data rectangles to index in the location index, the corresponding object index would need to index $N_O = N_L \cdot N_{static}$ rectangles as the number of objects to be indexed is $N_{static}$ times the number of locations to be indexed by the location

index. The density of data rectangles at level $l$ for the object index would also be approximately $N_{static}$ times the density of data rectangles for a corresponding location index at level $l$ which is denoted by $D_L^l = D_l$. Let $\lambda_{LI}$ denote node accesses for the location index and $\lambda_{OI}$ denote the number of node accesses for a corresponding object index. Then we get,

$$\begin{aligned}
\lambda_{LI} &= \sum_{l=1}^{h-1}(D_l + \frac{q_1 \cdot q_2 \cdot N_L}{f^l} + (q_1 + q_2) \cdot (D_l \cdot \frac{N_L}{f^l})^{\frac{1}{2}}), \\
\lambda_{OI} &= \sum_{l=1}^{h-1}(D_l \cdot N_{static} + \frac{q_1 \cdot q_2 \cdot N_L \cdot N_{static}}{f^l} \\
&\quad + (q_1 + q_2) \cdot (D_l \cdot N_{static} \cdot \frac{N_L \cdot N_{static}}{f^l})^{\frac{1}{2}}).
\end{aligned}$$

Calculating the ratio for node accesses,

$$\frac{\lambda_{LI}}{\lambda_{OI}} = \frac{1}{N_{static}}. \tag{5}$$

The above estimation assumes uniform distribution of data. We can also determine the number of tuples accessed for result retrieval for a location index and a corresponding object index. The number of leaf nodes accessed by the R-tree structure of location index $L$ can be derived using $l = 1$ in the equation for $\lambda_{total}(L, q)$ as,

$$\lambda_{leaf} = (D_L^1 + \frac{q_1 \cdot q_2 \cdot N_L}{f} + (q_1 + q_2) \cdot (D_L^1 \cdot \frac{N_L}{f})^{\frac{1}{2}}).$$

Due to the mapping of spatial locality of data to locality of data on disk in the location-centric model, we assume that all tuples corresponding to data rectangles on a single leaf node can be accessed using a single disk access by the location index. This forms a lower bound on the number of disk accesses required by the location-centric model to fetch relevant tuples from disk. Also assume that each tuple corresponding to a rectangle in the object-based model requires a single disk access. Number of tuples to be fetched by the object-based model is $f.f_{oc}$ times the number of node accesses at the leaf level as each leaf

node contains $f$ data rectangles on average, where $f$ is the fan-out for the object index. $f_{oc}$ is the fraction of data rectangles at leaf level that overlap the query window $q$; the tuples corresponding to these data rectangles satisfy the spatial constraints for the query and need to be fetched from disk for checking against other query constraints. This forms an upper bound on the number of disk accesses required here as buffering will lead to retrieval of some relevant tuples with a single disk access. Let $\psi_{LI}$ be number of disk accesses required by location index to retrieve relevant tuples from disk and $\psi_{OI}$ be number of disk accesses required by object index to retrieve relevant tuples from disk. Using the above assumptions we get

$$\psi_{LI} \geq (D_1 + \frac{q_1 q_2 N_L}{f} + (q_1 + q_2)(D_1 \frac{N_L}{f})^{\frac{1}{2}}),$$

$$\psi_{OI} \leq f f_{oc}(D_1 N_{static} + \frac{q_1 q_2 N_L N_{static}}{f} + (q_1 + q_2)(D_1 N_{static} \frac{N_L N_{static}}{f})^{\frac{1}{2}})$$

The ratio of disk accesses for a location index $\psi_{LI}$ compared to the disk accesses for its corresponding object index $\psi_{OI}$ is,

$$\frac{\psi_{LI}}{\psi_{OI}} \geq \frac{1}{f \cdot f_{oc} \cdot N_{static}},$$

where $f_{oc}$ is the fraction of data rectangles at leaf level that overlap the query window $q$; the tuples corresponding to these data rectangles satisfy the spatial constraints for the query and need to be fetched from disk for checking against other query constraints.

In conclusion, our analytical estimates suggest that the location index clearly performs much better than a corresponding object index. Our experimental evaluation in Section 2.6.2 corroborates the analysis provided here.

The size of location index will be significantly less than the size of a corresponding object index where static and moving objects are indexed together, since the location index needs to index fewer data rectangles. The node accesses for searching data in the index and for performing disk accesses to retrieve tuples from disk are also lower for the location index compared to an object index.

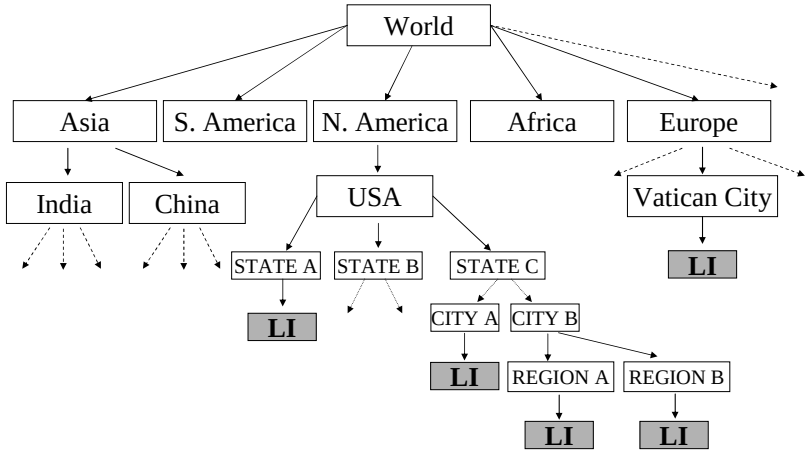



**Figure 4:** World Wide Location Index

### 2.4.4 Engineering a World Wide Location Index

The location indexing approach proposed above can be used to index all relevant locations of interest in the entire world. This requires engineering modifications to enable the system to handle queries efficiently. It is impossible to handle all locations of interest in a single LI; multiple LIs are required to handle the vast magnitude of location-dependent data that is generated. In order to answer queries efficiently, it is important to split a World Wide Location Index (WWLI) into smaller LIs. This is handled by engineering the WWLI as a hierarchical structure where we index locations with increasing resolution in a hierarchical manner.

The root node of the WWLI represents all relevant locations of interest in the entire world. The next level in this hierarchical structure represents the different continents which are further divided according to the geographical extent of countries. Some countries may be small enough to enable us to index all locations of relevant interest in a single LI. In the example displayed in Figure 4, *Vatican City* will be such a location. Larger countries will be divided according to natural geographical boundaries; for example, USA can be further represented by the states at the next level of the WWLI. The geographical division can be further performed by splitting states into counties, cities and so on till we are able to identify a region with has a reasonable number of locations of interest which can be inserted

in a single LI. In order to direct queries to the appropriate LI, we maintain a hashmap which stores a hash of the path to the LI beginning at the root node of the hierarchical structure (the *World* node). Each hash key for a particular path is mapped to the root node of the LI for the end point of the path. For example, the path $World \rightarrow Europe \rightarrow VaticanCity$ is hashed and stored as a key in our hashmap. The value for this key points to the database server which stores the LI for the region *Vatican City*. This mapping enables us to direct all queries to the appropriate LI based on the *region of interest* for the query.

## *2.5 Location Query Evaluation*

In this section, we describe our approach for evaluating different types of location queries, simultaneously discussing the concepts associated with each situation. We provide evaluation techniques for two representative types of location queries: *Moving Location Queries over Static Objects* and *Moving Location Queries over Moving Objects*. In order to handle moving queries, updating positions of mobile objects is essential for the precision and freshness of query results. However, frequent updates to the database server are expensive in terms of both communication costs and CPU and Disk IO costs for update processing and index maintenance at the database server. A number of techniques have been proposed to handle the location update of mobile objects, which attempt to balance the contrasting requirements of precision of query results and limited bandwidth usage.

A naïve approach for updating positions of mobile objects is the *periodical update* of object positions in which each moving object reports its current position after an interval of time. However, this approach suffers from low precision. Moreover, the approach also leads to a heavy load on the database server as motion updates to the server may need to be synchronized in order to compute query results consistently. Modeling motions of the moving objects for predicting their positions is another commonly used method in moving object indexing [68]. Motion modeling uses approximation for predicting the position of a moving object based on its available motion parameters using techniques such as *dead*

*reckoning*. The disadvantage with dead reckoning arises from the fact that the method is based on estimation, each object needs to sample its motion parameters at regular intervals and may not necessarily report all updates as soon as they occur. Another problem with these methods is that they might exclude some relevant results.

### 2.5.1 Moving Object Indexing

Updating positions of mobile objects is essential for the precision and freshness of query results in a mobile environment. This requires that each mobile object communicate its position and other aspects related to its motion modeling (like velocity, direction etc.) to the database server as frequently as possible for query results to be computed accurately. However, frequent updates to the database server are expensive as it involves communication from the mobile object to the database server. Secondly, the database server is required to update the entry corresponding to this mobile object in the object table and corresponding index. A large number of techniques have been proposed to handle the update of mobile objects which attempt to balance the contrasting requirements of precision of query results and limiting the usage of bandwidth.

A simple approach which can be adopted to update positions of mobile objects is a *periodical update* of the position parameters of the object where each object reports its current position after an interval of time $t_u$. The disadvantage associated with this approach is that query results may not necessarily be precise. The system has the position parameters for the moving object at a particular time and any change in the position of the object from the last update may invalidate the results for any queries involving these objects. This approach also leads to a heavy load on the database server as motion updates to the server need to be synchronized so that fresh query results can be computed after each update period.

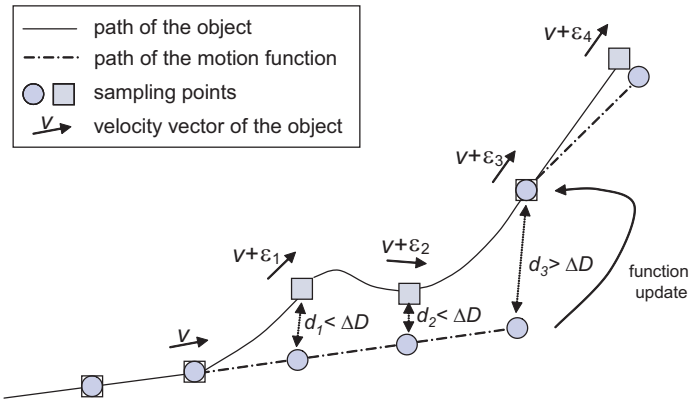



**Figure 5:** Dead reckoning

Modeling motions of the moving objects for predicting their positions is another commonly used method in moving object indexing [109, 68]. Motion modeling uses approximation for predicting the position of a moving object based on its available motion parameters. Instead of reporting their position updates each time they move, moving objects report their velocity vector and position updates only when their velocity vectors change *significantly*. This technique is known as *dead reckoning* [47].

At each time step a moving object samples its current position and calculates the difference between its current position and the position predicted by the dead reckoning algorithm based on the last motion update it reported to the server. In case this difference is larger than a specified threshold, say $\Delta D$, the new motion function parameters are relayed to the server. Figure 5 provides an illustration.

The disadvantage with dead reckoning arises from the fact that the method is based on estimation, each object needs to sample its motion parameters at regular intervals and may not necessarily be able to report the updates as soon as they occur. The bigger problem with these methods is that they might exclude some relevant results. We want to ensure that all relevant results are returned when evaluating location queries; a larger result set having some irrelevant results may be acceptable as long as all relevant results are present in the result set.

We now discuss a *grid-based approach* for updating the positions of moving objects

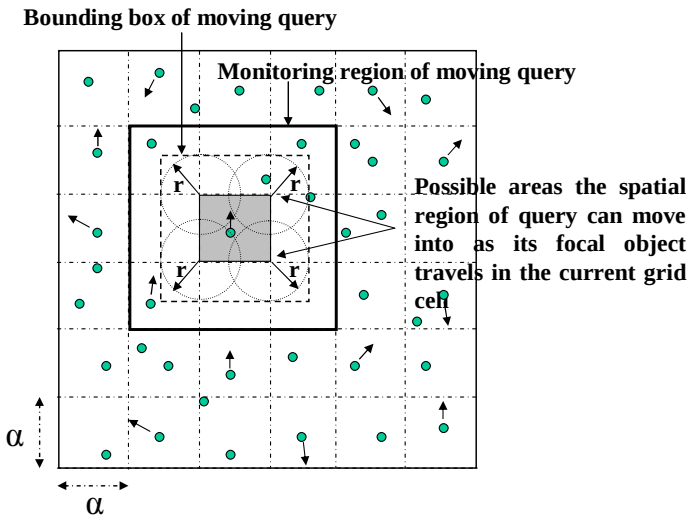



**Figure 6:** Grid-based model for moving objects

which attempts to deal with the above shortcomings while still providing only periodic position updates to the server. The grid-based framework proposes a simple approach for handling position updates of moving objects. Under this approach, the Universe of Discourse is split into a number of cells of side $\alpha$, with each moving object mapping its current position to a particular grid cell. Each moving object is responsible for initiating a position update when it moves from one $\alpha$-cell to another. The database server is only aware of the current grid cell for each moving object and computes query results involving moving objects using the current $\alpha$-cell as the position for each moving object. Figure 6 illustrates the grid-based approach for modeling motion parameters of moving objects. The possible areas the spatial region of a query may move into as the focal point of a moving query moves in its current grid cell is displayed in the figure.

As can be seen from the figure, the query bounding box calculated by this $\alpha$-cell model will be larger than the bounding box for the query obtained using the actual position of the moving object. For queries over static objects, the query result will consist of all *static* objects associated with locations lying inside the bounding box; whereas for queries over moving objects the monitoring region consists of all $\alpha$-cells that intersect with the bounding box for the query. This approach will provide a larger set of objects than the result set

obtained using the actual position of the object, but it guarantees that all relevant results will be present in this larger set of objects.

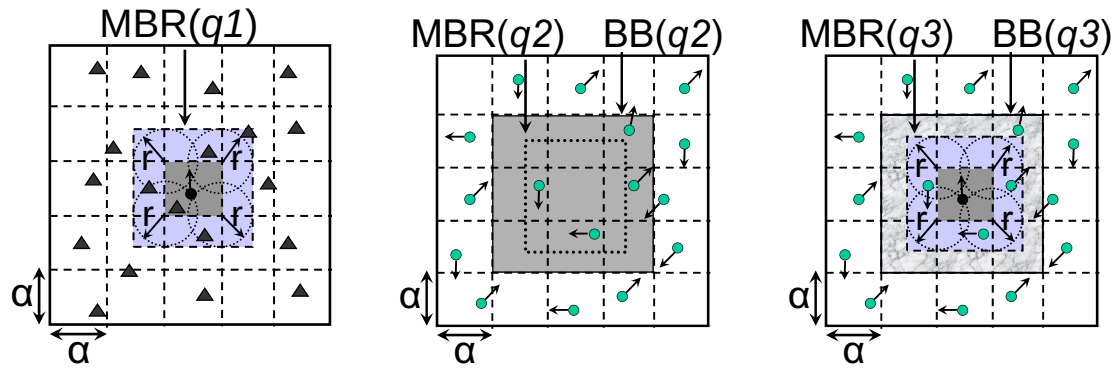
The parameter $\alpha$ determines the bounding box for a moving query and the monitoring region for queries over moving objects. Large values of $\alpha$ will lead to larger bounding boxes and consequently larger number of objects being included in the query results. Smaller values for $\alpha$ will lead to smaller $\alpha$-cells and more frequent updates to object positions as objects will be changing cells more often. Hence, it is important to use a suitable value for $\alpha$ taking both the above factors into account.

The advantages associated with the grid-based approach are: (1) It does not make any assumptions regarding the motion parameters of moving objects. (2) The approach is guaranteed to return at least all relevant results for a location query and updates the results as motion updates are received. (3) Motion updates are asynchronous, that is, each object triggers updates when it moves from one cell to another and as the updates do not occur at the same time the load on the database server is distributed. Due to these associated advantages, we use this grid-based framework to model positions of moving objects.

### 2.5.2 Query Evaluation Procedure

We now briefly describe the evaluation procedure for different types of location queries.

**Moving Query over Static Objects:** The procedure for evaluation of moving queries over static objects is illustrated by an example displayed in Figure 7(a). A location query *q1* is associated with a focal object; the position of this focal object is denoted using the current $\alpha$-cell in which it is located. This $\alpha$-cell is displayed using a dark grey cell in the figure. As the actual position of the object may lie anywhere inside this $\alpha$-cell the bounding box for the query comprises of the MBR for circles with radius *r* (radius of query) drawn at the four corners of the $\alpha$-cell. The light grey area in the figure displays the bounding box for the query *q1*. Any static objects associated with locations lying inside this bounded region will form the answer for the query. As long as the focal object of the query remains inside




(a) Moving Query Static Objects

(b) Static Query Moving Objects

(c) Moving Query Moving Objects

**Figure 7:** Location Query Evaluation

its $\alpha$-cell, the query result will remain unchanged. When the focal object moves to another cell, the query results need to be revised as the bounding box for query *q1* will change. Note that the bounding box and MBR for the query are the same in this case.

**Static Query over Moving Objects:** Figure 7(b) displays the procedure for evaluating static queries over moving objects. The bounding box for the static query *q2* is a rectangular region as shown in the figure. As the query is static the bounding box for the query remains the same. The monitoring region of the query consists of all $\alpha$-cells that intersect the bounding box of query *q2* as shown by the shaded area in the figure. Any moving objects lying within these $\alpha$-cells are potential candidates for answering the query and may be returned as the results for the query *q2*. Each $\alpha$-cell is associated with a set of queries; this set comprises of all queries that have monitoring regions intersecting with this $\alpha$-cell. When an object moves from one $\alpha$-cell to another the query results for queries associated with both the $\alpha$-cells are updated. Note that as the query is static, the set of static queries associated with any $\alpha$-cell remains the same and does not change over time; only the set of query results need incremental reevaluation as the objects move in or out of the associated $\alpha$-cells.

**Moving Query over Moving Objects:** The procedure for evaluating moving queries over moving objects is illustrated by example in Figure 7(c). The MBR for location query *q3* in

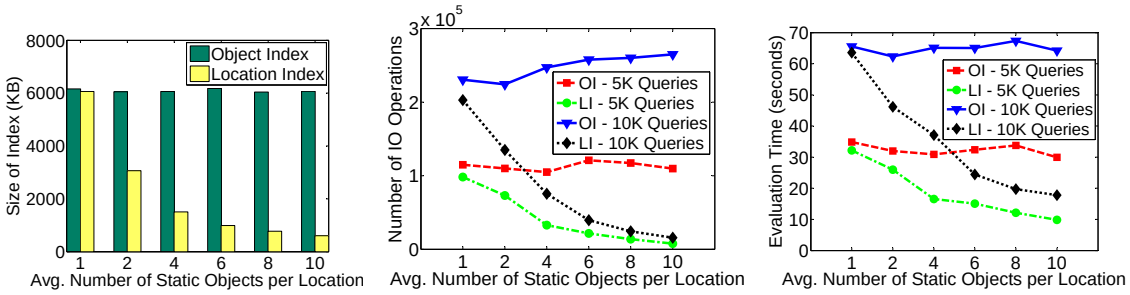
Figure 7(c) is defined in a similar manner as the MBR for location query *q1* in Figure 7(a). The bounding box $BB(q3)$ is the set of $\alpha$-cells intersecting the MBR of the query. Any moving objects lying within the bounding box are potential candidates for answering the query and may be returned as the results for the query *q3*. Each $\alpha$-cell is associated with a set of queries that have their bounding box intersecting with this $\alpha$-cell. When an object moves from one $\alpha$-cell to another the set of queries associated with both $\alpha$-cells may need to be updated. For each moving query over moving objects, the set of moving objects residing in its bounding box constitute the result of the query.

## *2.6 Experimental Evaluation*

This section describes three sets of experiments for evaluating the performance and effectiveness of our location index framework. We first describe the experimental setup adopted by us to evaluate the performance of location-centric modeling and indexing. The first set of experiments analyzes the behavior of the location index. The second set of experiments exhibits the advantages of location indexing over object indexing for evaluating queries over static objects. The third set of experiments considers a realistic environment comprising of static and moving queries over a set of static and moving objects. The performance of our location-centric framework against traditional indexing schemes is evaluated through a comprehensive study of the performance of the different approaches.

### 2.6.1 System Parameters and Setup

For all our experiments, we consider the Universe of Discourse to be a rectangular region expanding around 500,000 sq. miles. Moving or static queries over moving or static objects are considered in each scenario. Moving queries are assigned range values from the list $\{1, 2, 3, 4, 5\}$ miles using a Zipf distribution with parameter 0.6. Similarly, static queries are assigned side range values from the list $\{2, 3, 4, 6, 8\}$ miles using a Zipf distribution with parameter 0.6. The above mentioned parameters and default object densities closely follow previous work in the area; objects and queries are randomly distributed over the Universe of




(a) Index size (b) Number of IO Operations (c) Query Evaluation Time

**Figure 8:** Comparison of Location Index and Object Index Performance

Discourse. Moving objects follow random paths, each motion update will lead to a random direction and random speed being chosen for the object, with object speeds categorized into different values. We consider different classes of moving objects like pedestrians (0-5 miles/hour), slow moving vehicles (30-60 miles/hour) and fast moving vehicles (70-100 miles/hour). As each object is responsible for initiating an update when it moves from one $\alpha$-cell to another, no information regarding the motion of the object is required for query evaluation. Both the location index and object index are R-tree based indices, with a 100 *page* LRU buffer, each page 4 KBytes in size. Internal tree nodes have a branching factor of 100 with a fill factor of 0.7 in order to optimize performance [55].

### 2.6.2 Location Index vs. Object Index

In this set of experiments we study the advantages of location indexing (LI) over traditional object indexing (OI), as predicted by our analytical estimates in Section 2.4.3. Figure 8(a) plots the size of an OI and the size of the corresponding LI. The simulation setup involves a universe of discourse (UoD) containing 100K static objects. The distribution of static objects in the UoD is varied so that the average number of static objects per location ($N_{static}$) increases from one to ten. As can be seen in Figure 8(a) increasing $N_{static}$ does not affect the size of OI as it still needs to index 100K data rectangles. On the other hand, the size of LI decreases with increasing $N_{static}$; the number of rectangles to be indexed by LI decreases, as LI only needs to index (100K/$N_{static}$) rectangles. Figure 8(b) plots the total

IO operations required by both indices as we vary the average number of static objects per location. We also vary the number of queries (5K-10K) over the static objects. As can be observed from the figure LI, performs better than a corresponding OI over the static objects. Even with a single object per location, the location-centric approach performs better than object-based approach as retrieval of tuples from the relational table is optimized for the location-centric approach. This is due to the spatial locality of data being mapped to locality of data on disk. For $N_{static} = 1$, better organization of data on disk is solely responsible for superior performance of LI. As $N_{static}$ increases, the advantage associated with the LI further increases as search operations are carried out over a smaller index in case of LI. In fact, with ten objects per location on an average, LI requires only around 6% of the number of IO operations required by OI. As the number of objects remains the same throughout the experiment, the performance of OI almost remains the same as we vary $N_{static}$. Our simulator can provide approximations for disk IOs for relational data access which is included in the number of IOs. Further experiments evaluate the performance of LI against OI without considering relational data access. The evaluation times for the same scenario, as shown in Figure 8(c), show that evaluation over LI is much faster than evaluation over OI for static objects (except for $N_{static} = 1$). Again the query evaluation times for the OI remain almost constant as we vary $N_{static}$. Note that these plots are generated after averaging the results over a number of runs. Some oscillations in the results (for OI) are to be expected as our simulator simulates a different set of queries each time a UoD is generated.

### 2.6.3 Location Index Performance

Figure 9 plots the query evaluation times for a set of queries over the location index as the size of the index increases. The size of the location index is determined by the number of rectangles indexed which depends on the number of relevant locations in the *UoD*. As for any index structure, the query evaluation performance for the location index declines as the size of the index increases (Figure 9).

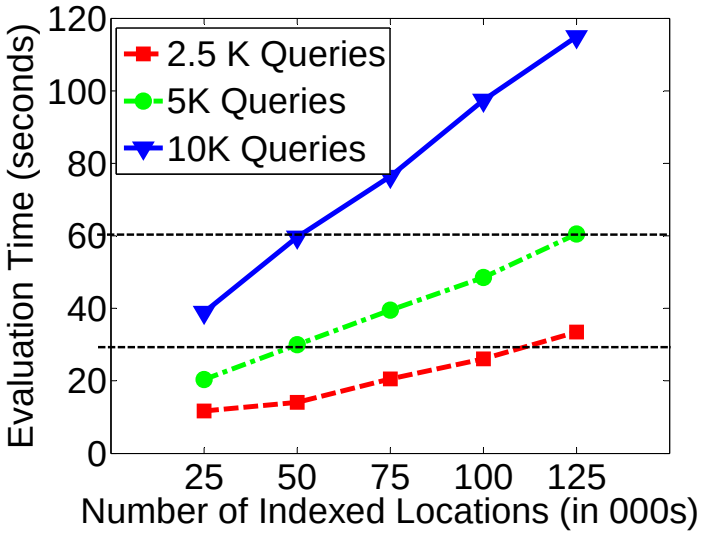



**Figure 9:** Evaluation Time vs. Size of Location Index

The horizontal lines at $t = 30$ sec. and $t = 60$ sec. help us benchmark the index performance. Depending on the average number of expected queries the service provider may guarantee a certain *Quality of Service(QoS)*, measured in terms of minimum query evaluation intervals at which user queries can be answered. For example, as can be seen from the figure, for a *QoS* guarantee of $t_s = 30$ sec., if the service is expected to handle 2.5K queries on average, then this location index can index around 115K locations. However, if the service is expected to handle 5K queries, the maximum number of locations that can be indexed falls to around 50K.

### 2.6.4 Location Query Performance Evaluation

Now we consider a scenario involving static and moving objects and display that the location-centric framework outperforms object-based modeling and indexing for a mixed workload comprising of moving and static queries. We compare the performance of three different approaches in this experiment. The first approach is the traditional object indexing approach, which requires all static and moving objects to be indexed as an object index. The second approach indexes locations of all static objects using the LI that we have developed; moving objects are still indexed using a traditional OI. We refer to this approach

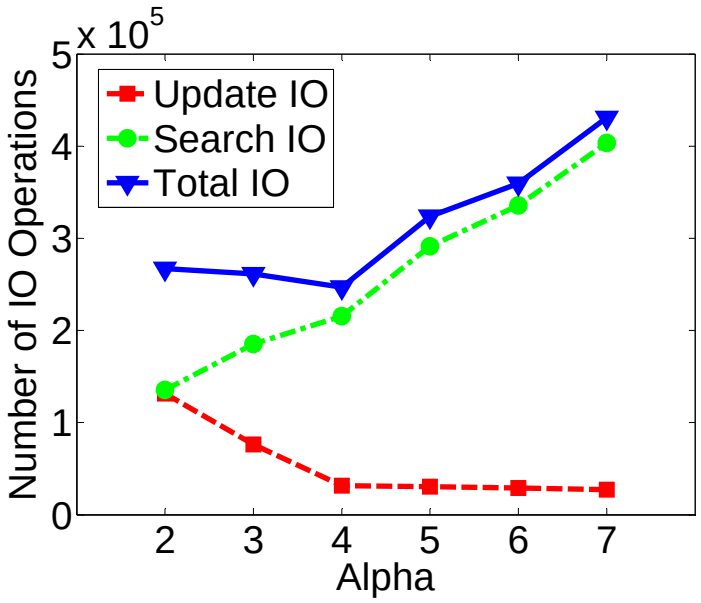


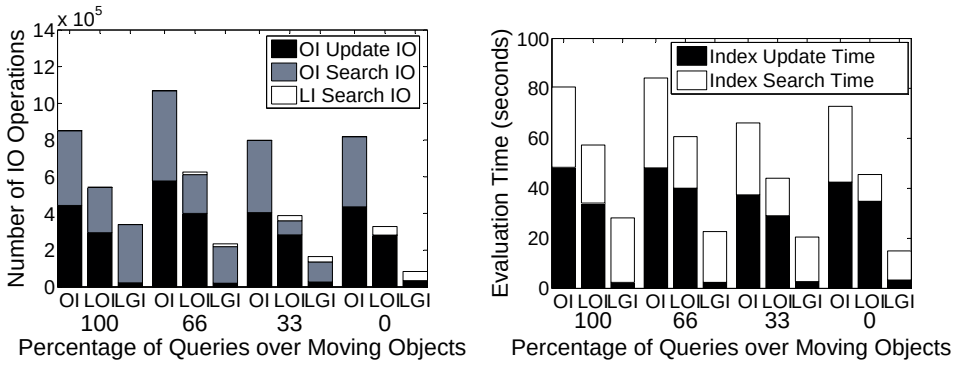

**Figure 10:** IO Costs with Varying $\alpha$

as the $LOI$ approach. Our third approach indexes locations for all static objects as a LI and maintains an OI over positions of moving objects using the grid-based framework described in section 2.5. We call this approach the $LGI$ approach. For this set of experiments, we perform query evaluation for a set of 20K queries over a UoD having 100K (50% static and 50% moving) objects.

#### *2.6.4.1 Determining $\alpha$*

The parameter $\alpha$ determines the size of the grid cell for approximating positions of moving objects. It is important to determine the optimal value of $\alpha$ for efficient system performance. Figure 10 displays the update, search and total IO costs for different values of $\alpha$.

For this experiment half of the queries are over moving objects and the other half over static objects. The following conclusions can be reached from Figure 10. As $\alpha$ increases the update IO cost for the system decreases; larger values of $\alpha$ imply objects will have to travel greater distances to shift $\alpha$-cells. For larger $\alpha$ values, fewer objects are expected to shift $\alpha$-cells for any time interval, thus leading to fewer updates. Similarly, the search IO




(a) Number of IO Operations (b) Query Evaluation Time

**Figure 11:** Query Evaluation Performance

costs rise with increasing $\alpha$ values due to larger bounding boxes as explained earlier. $\alpha$ is set to the value which balances the update IO and search IO costs thus providing lowest total IO costs. For the current set of experiments, we can observe that $\alpha = 4$ is the ideal value.

#### *2.6.4.2 Performance Evaluation*

Figure 11(a) plots the number of IO operations required for query evaluation, as we vary the fraction of queries over moving objects and explore the performance of the three indexing approaches as discussed above. The corresponding query evaluation times for all approaches are shown in Figure 11(b). Among queries over moving objects, half of the queries are static queries over moving objects and the other half are moving queries over moving objects. The IO operations consist of three different components: (a) object index update IO for moving objects, (b) object index search IO and (c) location index search IO. The $OI$ approach has just the first two components as it does not support a location index. As can be observed from the figure $LOI$ approach works much better than the $OI$ approach and the $LGI$ approach outperforms both the $OI$ and $LOI$ approaches. Even when all queries are over moving objects, $LOI$ approach works better as the object index for moving objects in the $LOI$ approach is smaller than the object index of the $OI$ approach.

This is because the object index of the $OI$ approach indexes static as well as moving objects whereas the object index in the $LOI$ approach indexes only the moving objects. As only half the objects are moving objects, this index is roughly half the size of the index of the $OI$ approach. The search component of the IO for the $LOI$ approach is much lower than the search IO required by the $OI$ approach. Further, as the percentage of queries over moving objects decreases, the difference in search IOs required by $LOI$ approach and the $OI$ approach increases. As all queries over static objects are directed to the location index, the search performance of the $LOI$ approach compared to the $OI$ approach improves as more queries are directed to the location index. $LOI$ approach has lower update costs too as updates are required over a smaller index compared to the $OI$ approach. The $LGI$ approach, further improves the associated update costs by adopting a grid-based framework for handling moving objects. The search costs for the $LGI$ approach are a little higher than those for the $LOI$ approach. This is simply due to the fact that the bounding boxes for the queries and MBRs for moving objects using the grid-based framework are larger than the corresponding query bounding boxes and MBRs based on exact positions of moving objects. Hence, depending on the value of $\alpha$, the result sets in the $LGI$ approach are a little larger than the result sets in the $LOI$ approach.

## *2.7 Related Work*

In this section we discuss previous work related to indexing and querying in a dynamic mobile environment relevant to our work. Location queries are tools for monitoring dynamically changing information whether it be content on the internet [77] or even streaming data [23]. Location queries over spatio-temporal data pose different challenges due to the multi-dimensional nature of the data involved as indices need to be built over multi-dimensional object data, where objects may be static or moving.

Research on object indexing in a mobile environment has been focused either on indexing current positions of moving objects [57, 94, 68, 50] or indexing the trajectories of

moving objects [92, 89, 71]. R-tree and its variants are the most commonly used indexing structures for spatio-temporal indexing of mobile objects. Indexing moving object positions poses problems due to frequent updates to the object positions. To deal with this problem [93] proposes indexing queries instead of objects for evaluating static location queries over moving objects. Others have adopted this framework for moving queries too and observed its benefits while evaluating few queries over a large number of mobile objects. Some work attempts to leverage the advantages associated with object indexing and query indexing by using both types of indexing to perform query evaluation [50]. The cost of maintaining two indices is justified by gains achieved in query evaluation.

Work has also been done to introduce new indexing structures like the TPR-tree [94], B+-tree based indexing [60], trajectory-based indexing [89], and to make the R-tree more update efficient [72]. However, all research exclusively focuses on update and indexing for mobile objects. Static objects are simply considered to be a special case of moving objects where its velocity is zero, and thus are treated in a similar manner as moving objects in most of the literatures to date. Our location index and location-centric framework exploits the performance benefits of separating static objects from moving objects and exhibits significant performance gains over conventional object-centric approaches.

We focus on a realistic environment comprising of static and mobile objects and adopt a divide and conquer approach to separate the modeling and indexing procedure for static and mobile objects. Most of the work on location queries over spatio-temporal data deals with either static location queries over moving object positions [93, 62, 68, 35, 98, 110] or moving location queries over static object positions [102, 98]. Some work deals with the issues involved in processing moving location queries over moving object positions [83, 50]. Work dealing with moving queries and moving objects assumes a similar platform can efficiently handle static queries and static objects. We consider all possible kinds of location queries and develop a location-centric framework for efficiently answering the different types of location queries.

Recent work concentrates on incremental evaluation of query results; SINA [83] and SEA-KNN [113] employ hashing based indexing techniques for both objects and queries and generate updates to previously evaluated results through a sequence of processes which involves hashing, invalidation of existing results and joining to produce new updates to existing query results.

Our incremental reevaluation is based on static $\alpha$-cells, with moving objects initiating position updates as they move across $\alpha$-cells, thus forcing reevaluation or incremental evaluation of associated queries. Most existing work concentrates on either range queries or kNN queries. [57] proposes a common framework to handle different types of spatial queries by developing location update strategies to suit the queries being monitored.

# CHAPTER III

# SCALABLE PROCESSING OF SPATIAL ALARMS USING SAFE PERIOD OPTIMIZATIONS

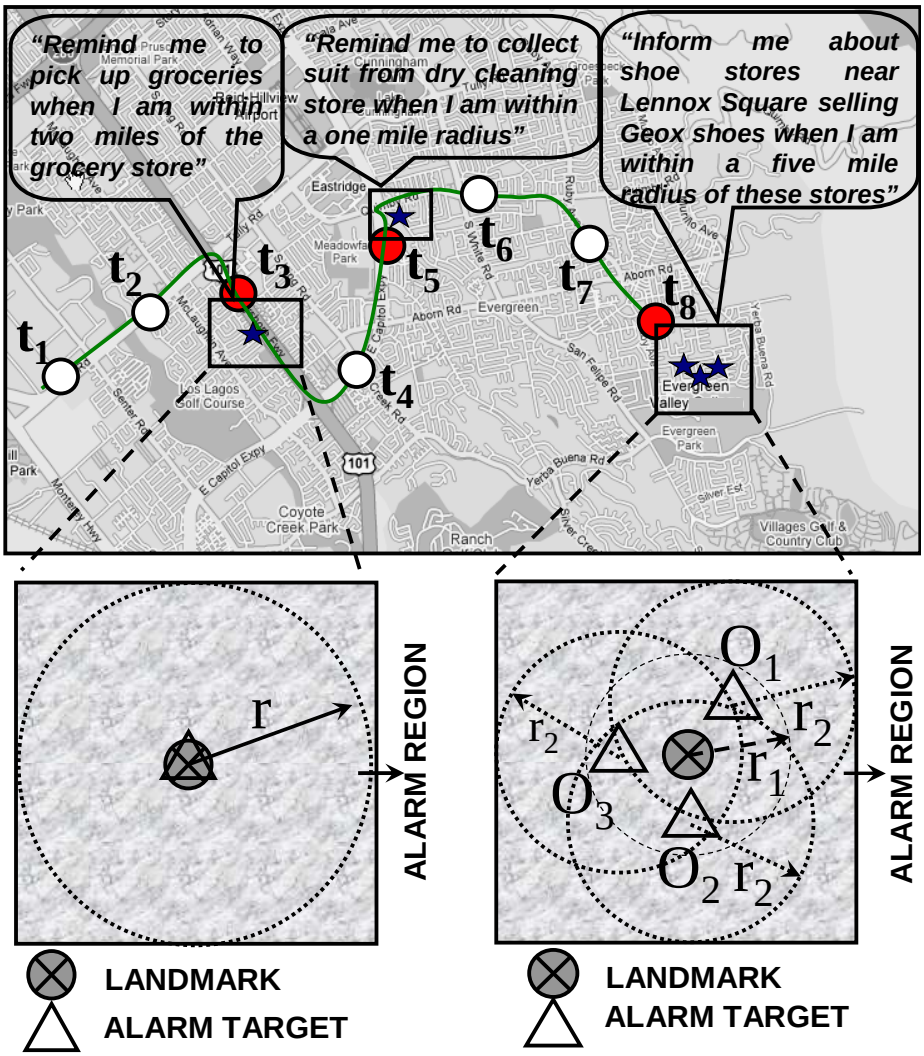
## *3.1 Introduction*

In today's fast-paced world, users need all available aid from existing technologies to make their busy lives simpler and organized in order to improve their efficiency. With the advent of mobile communication technology and continued price reduction of location tracking devices, location-based services(LBSs) are widely recognized as an important feature of the future computing environment [38]. In this work, we describe the spatial alarm application for LBSs and develop a scalable, efficient framework for handling the same.

Most people use time-based alarms in their daily lives in order to wake up in the morning or to remind them of important time-based events. Time based alarms are effective reminders of future events that have a definite time of occurrence associated with them. Spatial alarms extend the idea of time-based alarms to future events that do not have a definite time of occurrence but are known to be sensitive to spatial locations which mobile users may travel to in the near future. Just as time-based alarms are set to remind us of the arrival of a *future reference time point*, spatial alarms are set to remind us of the arrival of a *spatial location of interest*. Thus, spatial alarms can be modeled as location-based triggers which are fired whenever a mobile user enters the spatial region of the alarms. Spatial alarms provide critical capabilities for many mobile location-based applications ranging from real time personal assistants, inventory tracking, to industrial safety warning systems.

Figure 12(a) shows three spatial alarms installed on the region around the grocery store at the corner of Clairmont and Braircliff in Atlanta, the dry cleaning store near the house of the mobile user, and the sport shoe stores selling Geox shoes nearby or in Lennox Square.

(a) Spatial Alarms Example




(b) Single Alarm Target (c) Multiple Alarm Targets

**Figure 12:** Spatial Alarms

Assume that these alarms were installed at time instance $t_0$. Given user positions at future time points $t_1$ to $t_8$, as shown in the figure, the spatial alarms should be triggered at future time instants $t_3$, $t_5$ and $t_8$, informing the mobile user that she is entering the spatial alarm region of her specified location of interest.

Processing of spatial alarms requires meeting two demanding objectives: high accuracy, which ensures no alarms are missed, and high scalability, which guarantees that the alarm processing is highly efficient and scales to large number of spatial alarms and growing

base of mobile users. The conventional approach to spatial alarms involves periodic alarm evaluations at a high frequency. Each spatial alarm evaluation is conducted by testing whether the user is entering the spatial region of the alarm. High frequency is essential to ensure that none of the alarms are missed. Though periodic evaluation is simple, it can be extremely inefficient due to frequent evaluations of alarms and the high rate of irrelevant evaluations. This is especially true when the mobile user is traveling in a location that is distant from the spatial areas of all her location triggers, or when all her spatial alarms are set on spatial regions that are far apart from one another.

In addition, spatial alarms can be processed using server-based infrastructure or client-based architecture. Although the client component of the server-based approach to spatial alarm processing shares several capabilities with a client based approach, such as the map and text based installation of spatial alarms and notification of the triggered alarms, there are some key differences in terms of processing capabilities and optimization objectives between these two alternative architectures. A server-based approach must allow optimizations for processing spatial alarms installed by multiple mobile clients, whereas a client-based approach focuses more on energy-efficient solutions for evaluating a set of spatial alarms installed on a single client.

In this chapter, we describe a server-based approach to scalable processing of spatial alarms, aiming at optimizing the conventional approach of periodic alarm processing by advocating a motion-aware safe period-based alarm evaluation framework. Concretely, we formalize the concept of spatial alarms and the problem of spatial alarm processing. We introduce the concept of *safe period* to minimize the number of unnecessary spatial alarm evaluations, increasing the throughput and scalability of the system. We show that our safe period-based alarm evaluation techniques can significantly reduce the server load for spatial alarm processing compared to the periodic evaluation approach, while preserving the accuracy and timeliness of spatial alarms. Furthermore, we develop a suite of spatial alarm grouping techniques based on spatial locality of the alarms and motion behavior of

the mobile users, aiming at optimizing safe period computation at the server. We evaluate the scalability and accuracy of our approach using a road network simulator and show that our proposed framework for spatial alarm processing offers significant performance enhancements for the alarm processing server while maintaining high accuracy of spatial alarms, especially compared to the conventional periodic alarm evaluation approach.

Our centralized framework for spatial alarm processing is aimed at providing a scalable system with the ability to handle a large number of clients with large number of installed alarms. The strict QoS requirements of the system demand that no spatial alarms are missed. The hundred percent success rate requirement is essential as spatial alarms may come to the aid of the user in life threatening situations, for example, an alarm installed on a hazardous area by a traffic monitoring system may warn users of dangerous road situations. Such alarms fall under the *public* alarms categorization and are different from the *private* alarms as described in the above example scenario. We make three unique contributions in this work. Firstly, we develop a formal definition for the spatial alarm processing problem and describe a scalable, centralized architecture for handling the same. Secondly, we identify problems associated with spatial alarm evaluation using existing spatial querying frameworks. More concretely, we highlight the problems associated with a periodic, continuous evaluation of spatial alarms and provide a *safe period optimization* technique to handle the same. Our safe period optimization reduces the alarm processing time considerably; however, this approach presents another scalability problem associated with large safe period computation time. Thirdly, we suggest *attribute-based alarm grouping* techniques based on spatial locality of alarms, subscriber-specificity of alarms and nearest alarms-based grouping methods to rapidly reduce the safe period computation time. We also introduce subscriber mobility-based optimizations, which effectively combine with our alarm grouping techniques, to further lower safe period computation time. Extensive experimental evaluation is performed to justify the development of this framework and the various safe period optimization techniques.

The rest of the chapter is outlined as follows. Section 3.2 provides a formal description of the spatial alarm problem and describes the concepts associated with the same. Our *alarm-centric* spatial alarm processor architecture is also introduced here. After considering the various options for spatial alarm processing and their weaknesses, we describe the safe period computation technique in Section 3.3. Next we introduce various alarm grouping-based optimization techniques in Section 3.4 followed by a description of our subscriber mobility-based optimizations in Section 3.5. Our experimental evaluation results are detailed in Section 3.6 followed by a brief description of related work in Section 3.7.

## *3.2 System Overview*

In this section, we first define the concept of spatial alarms and formalize the problem of spatial alarm processing. Then we provide a discussion on different types of spatial alarms and give a brief overview of our server-based system architecture. In addition, we describe two alternative ways of processing spatial alarms discussing pros and cons of each, introduce the concept of safe period and discuss its benefits for alarm evaluation.

### 3.2.1 Spatial Alarms

A mobile user can define and install many spatial alarms. A spatial alarm is typically defined and installed by a mobile user and shared by many other users. We refer to the mobile users who define and install the spatial alarms as the publishers or owners of the alarms. The owner of an alarm may specify a list of potential mobile users with whom the alarm may be shared. The system will verify the interest of listed mobile users authorized to access the alarm and only those users who respond positively are subscribed to the alarm.

A spatial alarm is a location trigger with the following six basic components: *Landmark*, *Alarm Target*, *Alarm Region*, *Alarm Triggering Condition*, *Alarm Notification* and *Alarm Stop Condition*.

**Landmark (L)**: A landmark refers to a particular location reference which can be either a popular point of interest, such as the Eiffel Tower in Paris, or an area of interest such as a

university campus. The concept of a landmark is central to the definition of a spatial alarm as the alarm target objects, defined below, are in the vicinity of a landmark.

**Alarm Target (T):** Alarm targets are objects of interest which the subscribers of the alarm will travel to at a future time point. Typically, alarm target objects may be associated with some filters specifying the matching conditions on the alarm targets. A spatial alarm may have a single target object or multiple target objects.

**Alarm Region (R)**: Alarm region is defined as the area around the alarm targets *T* upon entering which the user requires the associated spatial alarm to be triggered. The spatial alarm region may be specified by an area of radius *r* around $T$ or any other spatial region of regular or irregular shape. The *minimum bounding rectangle R* is used to approximate all alarm regions for processing convenience; $R$ is denoted using the bottom-left and top-right corner points: $(x_{bl}, y_{bl})$ and $(x_{tr}, y_{tr})$ respectively.

Consider the third spatial alarm example in Figure 12(a). Lennox Square is the landmark of the alarm, the sport shoe stores that sell Geox shoes are the alarm targets, and area within a five mile radius to each shoe store of interest near Lennox Square is the alarm region. Figures 12(c) illustrates this example scenario, in which $r_1$ is used to measure the concept of nearby Lennox Square and $r_2$= 5 miles, the distance to the shoe stores. Some alarms may have a single target object which may be the same as the landmark for this alarm; the first alarm on grocery store in Figure 12(b) is an example of such an alarm.

**Location Trigger**: Each alarm has an associated location trigger, which defines the spatial condition to be monitored to determine if and when an alarm should be triggered. Location triggers can be implicit in the spatial alarm specification or be specified explicitly by the user.

For example, all three example alarms in Figure 12(a) have implicit location triggers set on the encounter point where the mobile user enters the alarm monitoring region. The system will set a default Euclidean distance condition between the mobile subscriber of an alarm and the alarm region, namely $distance(S_i, R) = 0$, for location trigger of alarm.

**Alarm Stop Condition**: Alarm stop condition can be specified by a future time point or a future time interval. We use the alarm stop condition to define the time interval during which the alarm is alive, denoted as $[t_s, t_e]$, where $t_s$ is the starting point of the alarm and $t_e$ is the termination point of the alarm. Each spatial alarm needs to define an alarm termination condition to allow the system to correctly remove the alarms.

**Alarm Notification**: Spatial alarms are not relevant to all moving objects (or users) in the system. Any alarm will have an associated owner and one or more potentially interested subscribers. In general, the alarm owner (installer of the alarm) may identify a list of potential subscribers who may be permitted by the system to use the alarm. The system determines the list of active subscribers for an alarm by seeking acknowledgements from these potential subscribers who may accept or reject the alarm. Alarm notification messages are sent to subscribers of the alarm upon alarm activation. Each notification consists of notification recipients, notification actions and notification methods. Notification actions may be simple, such as displaying the notification message on user device, or more complex, such as opening a grocery shopping list on the user device along with the notification. Notification methods can be real-time and interactive when the mobile client is active, or deferred when the mobile client is in sleep mode.

Based on these concepts we formally define the spatial alarm evaluation problem below.

**Formal Problem Definition:** For any spatial alarm *A,* given a *landmark L* at location $(x_l, y_l)$, *alarm targets T* around L, *alarm region R* covering an area of radius *r* around the locations of *T* and a set of interested *subscribers* $S = \{S_1, S_2, ..., S_n\}$, each spatial alarm evaluation determines the subset of subscribers: $\{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\} \epsilon S$, which enter the alarm region $R$ at any instant of time $t \epsilon [t_s, t_e]$, where $[t_s, t_e]$ indicates the duration of time for which the spatial alarm $A$ is active.

### 3.2.2 Spatial Alarm Categorization

We categorize spatial alarms based on two criteria: publish-subscribe scope of the alarms and motion characteristics of alarm targets and alarm subscribers.

**Categorization by Publish-Subscribe Scope:** We classify spatial alarms into three categories - *private*, *shared* and *public* alarms - based on the publish-subscribe scope of the alarms. *Private* alarms are installed and subscribed to exclusively by the alarm owner. *Shared* alarms are installed by the alarm owner with a list of $k$ ($k > 1$) authorized subscribers and the alarm owner is typically one of the subscribers. In contrast with the privacy desired by the owner of a private alarm, the owner of a *shared* alarm wishes to share the alarm with a specified list of system users. Alarm sharing may typically be limited to a few system subscribers. For example, the user in example 1 may decide to share her alarm with her entire household. The service provider disseminates the alarm information to all users listed by the owner. It is up to the users to accept or reject the alarm; the system determines the subscriber list based on user response. *Public* alarms are usually installed with the purpose of sharing them with all mobile users who are entering the spatial regions of the alarms and are subscribers of the alarms. Mobile users may subscribe to *public* alarms by topic categories or keywords, such as *"traffic information on highway 85North"*, *"Zagat survey top ranked local restaurants"*, to name a few. Without loss of generality, rest of the chapter assumes that public alarms are subscribed to by all users. Alarms indicating hazardous road situations or heavy road congestion are examples of alarms that fall under this category.

**Categorization by Motion Characteristics:** Spatial alarms may also be categorized based on the motion characteristics of alarm targets and alarm subscribers. We illustrate our system design in terms of three classes of spatial alarms by motion characterization. The first class is the *Mobile Subscribers Static Targets* (MSST) alarms, where alarm targets are typically set on still objects such as restaurants, hospitals, churches, office buildings, and so forth. The second class is referred to as the *Static Subscribers Mobile Targets* (SSMT)

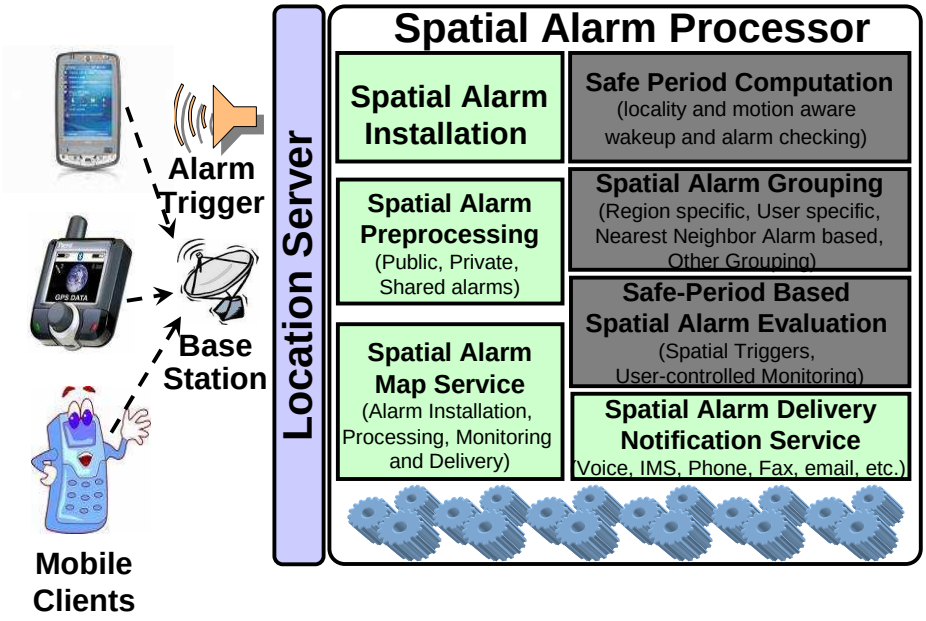



**Figure 13:** Spatial Alarm Processor Architecture

alarms, where alarm targets are moving but the position of mobile subscribers remains unchanged during the alarm validity period. A typical example of such alarms is *"tell me when the bus is within 2 miles of the bus stop near my office"*. The third class, where both alarm subscribers and alarm targets are moving, is called the *Mobile Subscribers Mobile Targets* (MSMT) alarms. A typical example of such alarms is *"inform me when my jogging buddies Amy and Josh are within a mile of my current location"*.

### 3.2.3 System Architecture

We assume that mobile users update their positions continually through either periodic location updates or dead reckoning or other location estimation techniques based upon known speed, elapsed time and course of movement. The proposed spatial alarm processing system architecture is shown in Figure 13, and it consists of three main components: *alarm installation or removal*, *alarm evaluation and optimization* and *alarm notification and delivery*.

The **alarm installation or removal** component accomplishes three main tasks: the installation of a spatial alarm from an authorized user, the specification of publish-subscribe

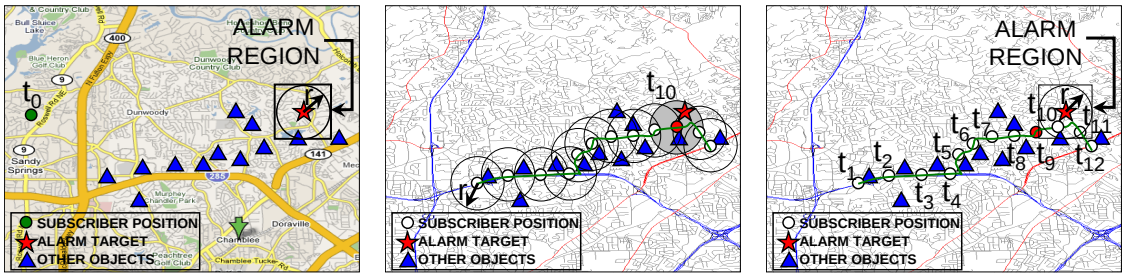
scope of the alarm, such as the authorized subscriber list, and the authorized removal of existing alarms. Only the alarm owner is authorized to delete the active alarms she has installed.

Upon the firing of a spatial alarm, the **alarm notification and delivery** component performs two major tasks. First, it informs the subscriber(s) who trigger(s) the alarm about the activated alarm by performing the set of actions associated with the alarm. Second, it checks if the alarm is one-time alarm or a continuous alarm with a specified duration. It removes the alarm from the active alarm queue if it is a one-time alarm. Otherwise, it triggers the alarm evaluation component to determine the alarm check period for periodic approach or to compute the safe period for our approach. Alarm notification methods may vary with different types of mobile devices depending on the latency constraint and the available means of delivery (voice message, text message, etc.).

The **spatial alarm evaluation** component works in three phases. First, it accepts spatial alarms as input and indexes them using the underlying spatial indexing structures during the alarm preprocessing phase. Next, the optimization phase applies alarm optimization techniques to produce a near-optimal alarm processing schedule. For example, safe period computation and alarm grouping are performed in the optimization phase. In the third phase, the actual alarm evaluation takes place. We call this phase run-time alarm execution. In this chapter we focus on the design of optimization techniques for spatial alarm evaluation.

### 3.2.4 Spatial Alarm Processing

We dedicate this section to discuss the weaknesses of existing spatial query processing techniques when applied to spatial alarm processing. Then we describe the advantage of our motion-aware safe period based alarm evaluation approach. We use a concrete example to facilitate the discussions. Figure 14(a) displays the map for Chamblee region of Georgia and an example alarm installed by a mobile user at time instant $t_0$ with the alarm region of




(a) Alarm Installation (b) Continuous Range Query Approach (c) Periodic Alarm Evaluation

**Figure 14:** Naïve Evaluation Techniques

radius *r* around the alarm target.

Figures 14(b) and 14(c) display the road network for the same region shown in Figure 14(a), extracted from the USGS [8] database, which we use for experimental evaluation. As shown in Figure 14(c), we consider a series of user positions collected from $t_1$ to $t_{12}$, each time instant reflects the subscriber's position at an interval of one minute. We first discuss how to process the installed alarm using existing spatial continuous query framework and show why spatial query processing techniques are inefficient when directly applied for processing spatial alarms.

The *spatial continuous query approach* would process the spatial alarm by transforming the alarm into a *user-centric* continuous spatial query. Given the alarm region of radius $r$ around the alarm target and the mobile user's current location marked by $t_0$, the transformed spatial query is defined by the query range $r$ with the mobile alarm subscriber as the focal object of the query. The evaluation of this spatial query proceeds at time instant $t_1$ and the query processor checks if the obtained query results contain the alarm target object. This process repeats periodically at each of the marked time instants until the alarm target is included in the query results at time instant $t_{10}$ (query region marked by shaded area in Figure 14(b)). The obvious drawback of this approach is the amount of unnecessary processing performed in terms of both the number of evaluations and the irrelevant query result computation at each evaluation. The farther away the mobile user is from her spatial alarms, larger the amount of unnecessary evaluations incurred using the spatial continuous

query approach.

A continuous query framework would require the service provider to periodically retrieve all static objects within a distance $r$ (alarm range) from the user location and check if the alarm target is included in the list of retrieved objects. The system treats the user information need as a continuous moving query over static objects which needs to be periodically evaluated as object position changes continuously from $t_1$ till $t_{10}$ when the alarm is triggered. At each position the static objects within the region of radius $r$ from the user location are retrieved only for the system to discover that they are irrelevant to this specific query. The alarm target is retrieved only at time instant $t_{10}$.

Alternatively, we can use a *periodic alarm evaluation* technique as shown in Figure 14(c). At each time instant from $t_1$ onwards, the system needs to determine if the current object position lies within the MBR of the spatial alarm region. In case alarm evaluation is performed at an interval of one minute, periodic spatial alarm processing would evaluate this condition periodically from $t_1$ to $t_9$ and trigger the alarm at $t_9$ as the subscriber reaches the spatial alarm boundary at this time instant. If the alarm evaluation period is changed to two minutes, the alarm trigger will be fired at $t_{10}$ instead of $t_9$. If the alarm evaluation period is set for four minutes, this alarm will be missed as the alarm evaluation takes place only at time instants $t_4$, $t_8$ and $t_{12}$ and at all evaluation times the subscriber is outside the alarm region.

Although the periodic evaluation does not incur irrelevant query result computation while processing spatial alarms, it suffers from a number of drawbacks. First, alarm miss rate is unpredictable as there is no appropriate technique for the system to determine the ideal alarm evaluation period. In case of high alarm miss rate the system fails to meet the high accuracy requirement of spatial alarm processing. Second, the periodic alarm evaluation approach is expensive as it performs a large number of unnecessary evaluations; hence, it is not scalable in the presence of a large number of alarms installed by a large number of mobile users. The amount of unnecessary evaluations increase as the mobile

users move farther away from their alarms.

Bearing in mind the problems inherent with the continuous spatial query evaluation approach and drawbacks of the periodic alarm evaluation approach, we develop a motion-aware safe period-based alarm evaluation approach. The goal of applying safe period optimization is to minimize the amount of unnecessary alarm evaluations while ensuring zero or very low alarm miss rate. The other technical challenge behind safe period optimization is to minimize the amount of safe period computation, further improving system scalability and achieving higher throughput. We describe our basic approach for safe period computation in the next section and address the challenge of minimizing the amount of safe period computations in Sections 3.4 and 3.5.

## *3.3 Safe Period Computation*

We outlined the problems associated with periodic alarm evaluation in the previous section. In this section, we discuss the safe period approach for determining an optimal schedule for alarm evaluation. Safe period computations help us meet the twin tasks of minimizing alarm evaluation load at the server as well as ensuring excellent Quality of Service (QoS) for system users. QoS goals associated with spatial alarm processing require that all alarms be successfully triggered for all alarm subscribers.

*Safe period* is defined as the duration of time for which it is safe not to check a particular alarm for a particular subscriber as the probability of this alarm being triggered for the subscriber is zero. Consider a subscriber $S_i$ and a spatial alarm $A_j$ ($1 \leq j \leq M$, $1 \leq i \leq N$), where $N$ is the total number of mobile users and $M$ is the total number of alarms installed in the system. The safe period of alarm $A_j$ with respect to subscriber $S_i$, denoted by $sp(S_i, A_j)$ can be computed based on the distance between the current position of $S_i$ and the alarm region $R_j$, taking into account the motion characteristics of $S_i$ and $A_j$.

Concretely, for alarms of the class Mobile Subscribers Static Targets, the two factors that influence the computation of safe period $sp(S_i, A_j)$ are (i) the velocity-based motion

characteristic of the subscriber $S_i$, and (ii) the distance from the current position of subscriber $S_i$ to the spatial region $R_j$ of alarm $A_j$. Thus the safe period $sp(S_i, A_j)$ can be computed as follows:

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_{S_i})} \tag{6}$$

Similarly, for Static Subscribers Mobile Targets alarm, the safe period $sp(S_i, A_j)$ is computed by taking into account (i) the distance from the current position of subscriber $S_i$ to the spatial region $R_j$ of alarm $A_j$, and (ii) the velocity-based motion characteristic of the mobile alarm target, using the following formula:

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_T)} \tag{7}$$

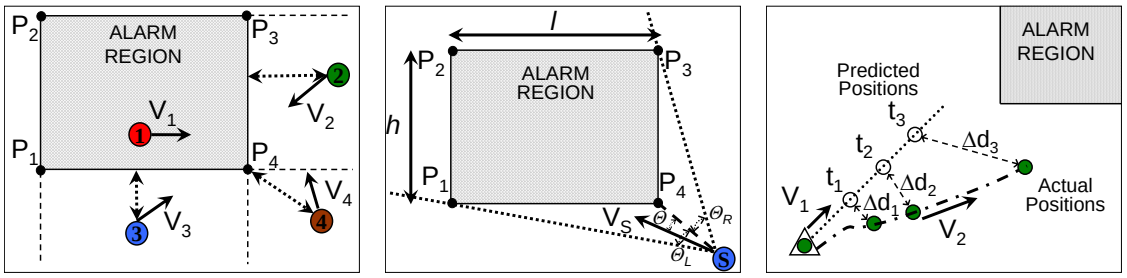
For spatial alarms of class Mobile Subscribers Mobile Targets, the motion characteristics of both subscriber and alarm target need to be considered for the computation of the safe period $sp(S_i, A_j)$, in addition to the distance between the current location of the mobile subscriber and the moving alarm region.

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_{S_i}, V_T)} \tag{8}$$

Clearly, the distance measure between the current location of the mobile subscriber and the moving alarm region $R_j$ is the first important parameter for safe period computation. The second important parameter is velocity measure of the mobile subscribers or the mobile alarm targets.

### 3.3.1 Distance Measurements

We use *Euclidean distance* approach as the basic distance measure for safe period computation. It measures the minimum distance from the current position of the mobile user, denoted as $P_m = (x_m, y_m)$, to the spatial alarm region $R$. Though the Euclidean distance measurement is simple, it may at times underestimate the safe period for a given alarm-subscriber pair.




(a) Euclidean Distance Measure (b) Steady Motion Assumption (c) Dead Reckoning

**Figure 15:** Basic Safe Period Computation

Consider a spatial alarm region $R$ covering the rectangular region represented by four vertices of a rectangle: $(P_1, P_2, P_3, P_4)$, as shown in Figure 15(a), where $P_1 = (x_1, y_1)$, $P_2 = (x_1, y_2)$, $P_3 = (x_2, y_2)$ and $P_4 = (x_2, y_1)$. The minimum Euclidean distance from $P_m$ to the spatial alarm region $R$, denoted by $d_{m,R}$, can be computed by considering the following four scenarios: ① when the mobile subscriber lies inside the spatial alarm region the distance $d_{m,R}$ is zero; ② when the mobile subscriber is within the $y$ scope of the spatial alarm region, the minimum euclidean distance is the distance from the mobile subscriber to the nearer of the two spatial alarm edges parallel to the x-axis; ③ when the mobile subscriber is within the $x$ scope of the spatial alarm region, minimum euclidean distance is the distance from the mobile subscriber to the nearer of the two spatial alarm edges parallel to the y-axis; and ④ when the mobile subscriber is outside both the $x$ and $y$ scope then the distance is the minimum of the euclidean distance to the four vertexes. Formally, $d_{m,R}$, the minimum Euclidean distance from mobile position $P_m$ to the spatial alarm region $R$, is computed using the following formula:

$$d_{m,R} = \begin{cases} 0, & x_1 \leq x_m \leq x_2 \\ & and \;\; y_1 \leq y_m \leq y_2 \\ min(|x_m - x_1|, |x_m - x_2|), & y_1 \leq y_m \leq y_2 \; only \\ min(|y_m - y_1|, |y_m - y_2|), & x_1 \leq x_m \leq x_2 \; only \\ min(d_{m,1}, d_{m,2}, d_{m,3}, d_{m,4}), & otherwise \end{cases}$$

$d_{m,1}, d_{m,2}, d_{m,3}, d_{m,4}$ denote the Euclidean distance from $P_m$ to the four rectangle vertexes

$P_1, P_2, P_3, P_4$ respectively. The distance function $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is used to compute the Euclidean distance between two points $P_i$ and $P_j$.

The safe period formulae in equations 1, 2 and 3 assume that the subscriber heads towards the alarm region in a straight line along the direction of the minimum Euclidean distance, an assumption that is rarely true in real life. One way to relax this stringent condition is to use the *steady motion assumption*: If the subscriber is heading towards the alarm region $R$, then the deviation in his motion direction is not likely to be extreme. Figure 15(b) shows a scenario where the bounded deviation in subscriber motion is taken into account for calculating average safe period for subscriber $S$ approaching alarm region $R$. In order for the subscriber $S$ to enter the alarm region $R$ at some future time instant, the average angle of motion for the subscriber $S$ over the safe period must lie between $-\theta_L$ and $+\theta_R$ (as shown in the figure), which we refer to as *alarm trigger angular range*. Assume that the mobile subscriber heads towards the alarm region $R$ in a direction at an angle $\theta$ from the minimum Euclidean distance vector; we refer to the distance from the subscriber position to the alarm region as the steady motion distance, denoted as $sm_{dist}(\theta)$. The steady motion-based safe period can be determined by $sm_{dist}(\theta)/f(V_S)$. Using the average steady motion distance obtained by computing $sm_{dist}(\theta)$ over all $\theta$ values ranging from $-\theta_L$ to $+\theta_R$, the steady motion-based safe period over the alarm trigger angular range can be calculated as,

$$sp = \frac{\int_{-\theta_L}^{+\theta_R} sm_{dist}(\theta) d\theta}{f(V_S) \int_{-\theta_L}^{+\theta_R} d\theta} = \frac{l + h}{f(V_S)(\theta_R + \theta_L)}, \tag{9}$$

where $l$, $h$ denote the length and height of the spatial alarm region. The steady motion assumption provides a more realistic and optimistic measure for safe period computations compared to the minimum Euclidean distance approach.

### 3.3.2 Velocity Measurements

**Maximum Speed:** The use of maximum travel speed of the mobile client for the velocity function $f(V_S)$ carries both advantages and disadvantages. On one hand, the 'maximum

travel speed' can be set by pre-configuration based on a number of factors, such as the nature of the mobile client (such as a car on the move or a pedestrian walking on the street), or the types of roads used. On the other hand, the maximum speed-based velocity estimation is often over pessimistic especially in the following two scenarios: (1) when the mobile client stops for an extended period of time; or (2) when the mobile client suddenly turns onto a road with very low speed limit.

Figure 15(a) shows the use of maximum speeds $V_2$, $V_3$ and $V_4$ for subscribers outside the alarm region to compute the safe period of the alarm. For dynamic alarms, a combination of the maximum speed for the alarm region $V_T$, as determined by the maximum target speed, and the maximum speed for the subscriber $V_S$ can be used to determine $f(V_S, V_T)$; one such possible function is $f(V_S, V_T) = V_S + V_T$. Note that this function is extremely pessimistic in calculating optimal safe period values as it assumes that the target and subscriber are directly heading towards each other in the direction associated with the minimum euclidean distance between them. Assessing the subscriber and target object velocities along the direction associated with the minimum euclidean distance would provide more optimistic measures for safe period especially when the target and subscriber are moving in a similar direction.

Another issue related to the use of maximum speed of a mobile client for the velocity function $f(V_S)$ is related to alarm misses. The maximum velocity based approach may fail to trigger alarms in cases where the maximum speed for the mobile subscriber increases suddenly. For example, a vehicle moving from a street onto a state highway would experience a sudden increase in its velocity, which may invalidate safe period computations. One way to address such sudden increase in velocity is to use *dead reckoning* techniques which require the mobile user to report to the server when her velocity increases over a certain threshold, as shown in Figure 15(c). The use of dead reckoning or similar techniques will allow the server to recompute the safe period for all alarms subscribed by this mobile client upon any significant velocity change.

In Figure 15(c), the mobile client keeps track of its predicted positions based on its maximum speed and its actual positions. As soon as the difference between the predicted position and the actual position exceeds a given threshold value (say $\delta$), the client provides its current speed $V_2$ to the server. If $V_2 > V_1$, where $V_1$ is the previously recorded maximum speed, the spatial alarm server uses the current reported speed $V_2$ to infer the type of road on which the user is traveling and the maximum travel speed for the road.

**Expected Speed:** One way to address the pessimistic nature of the maximum speed-based safe period computations is to use the expected speed for the velocity function. The future expected travel speed of a mobile client is computed as the sum of the current expected speed weighted by a factor $\alpha$ and the maximum speed weighted by a factor $(1-\alpha)$. Lower $\alpha$ values provide similar speed estimates as the maximum speed measure described earlier. Expected speed calculations are based on an *exponentially weighted average* over the current and previous location of mobile client (weighted by $\beta$) and previous expected speed calculation (weighted by $(1-\beta)$).

$$V^c_{expected} = \beta * \frac{D(l_c, l_p)}{(t_c - t_p)} + (1-\beta) * V^p_{expected} \tag{10}$$

$$V_{expected} = \alpha * V^c_{expected} + (1-\alpha) * V_S \tag{11}$$

where $V^p_{expected}$, $V^c_{expected}$, $V_{expected}$ are the previous, current and future expected travel speed of the subscriber, $t_c$, $t_p$ represent current and previous time instance for expected speed computation and $l_c$, $l_p$ represent the subscriber position at these time instances respectively.

### 3.3.3 Safe Period Based Alarm Evaluation

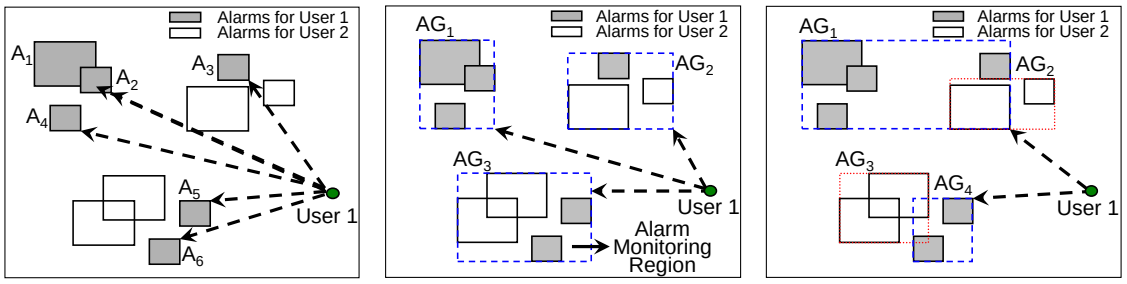
The safe period-based approach processes a spatial alarm in three stages. First, upon the installation of a spatial alarm, the safe period of the alarm with respect to each authorized

subscriber is calculated. Second, for each alarm-subscriber pair, alarm evaluation is triggered upon the expiration of the associated safe period and a new safe period is computed. In the third stage, a decision is made regarding whether the alarm should be fired or should wait for the new safe period to expire. If the new safe period is larger than a system-supplied threshold $t_\delta$, it means that the mobile client is still some distance away from the alarm region. However, if the new safe period is smaller than $t_\delta$, it means that the mobile client is entering the alarm region and the alarm is triggered.

When compared to periodic alarm evaluation, the safe period approach for spatial alarm processing reduces the amount of unnecessary alarm evaluation steps, especially when the mobile subscriber is far away from all her alarms. On the other hand, the main cost of the basic safe period approach described in this section is due to the excessive amount of unnecessary safe period computations, as the basic safe period approach performs safe period computation for each alarm-subscriber pair, regardless of the distance between the current location of the subscriber and the alarm region. Given $n$ users with an average of $m$ spatial alarms relevant to each user, the complexity of safe period computation is $O(n \cdot m)$. One obvious idea to reduce the amount of unnecessary safe period computations is to group spatial alarms based on geographical proximity and calculate safe period for each subscriber and alarm group pair instead of each alarm-subscriber pair.

## *3.4 Alarm Grouping Techniques*

The basic premise behind alarm grouping is to reduce the number of safe period computations while ensuring no alarm misses. In this section we present three alternative grouping techniques, each of which offers different degree of improvement for safe period computations. First, we group all alarms based on their spatial locality without considering subscriber specificity of the alarms. Alternatively, we apply spatial locality based-grouping to alarms of each individual subscriber. Both our analytical and experimental study show that this approach is more effective. The third locality-based alternative is to employ the




(a) Per Alarm Safe Period Computation (b) Spatial Locality-based Grouping (c) Subscriber-Specific Spatial Locality-based Grouping

**Figure 16:** Alarm Locality-based Grouping

nearest alarms-based grouping, which is effective but costly when there are frequent alarm additions and removals.

In addition to spatial locality based grouping techniques, we also develop subscriber mobility-based optimizations to further improve the scalability of alarm processing, which will be discussed in Section 3.5.

### 3.4.1 Spatial Locality-based Grouping

Spatial locality-based grouping considers the set of alarms from all users and inserts the nearby alarms into alarm groups. This approach outperforms basic safe period alarm evaluation if each group has a large number of alarms belonging to the same subscriber. Figure 16(a) displays the alarm regions for a set of installed alarms. The alarms for user 1 are marked by shaded alarm regions. Basic safe period evaluation computes the safe period for each of the six alarms $\{A_i \mid 1 \leq i \leq 6\}$ subscribed by the mobile user 1. In comparison, Figure 16(b) shows three groups derived from spatial locality-based grouping technique. We use a simple R-tree implementation in order to group alarms and identify the *minimum bounding rectangles (MBRs)* for alarm groups which are also referred to as *alarm monitoring regions*. Instead of computing safe period for each alarm-subscriber pair, spatial locality-based grouping requires the system to calculate a safe period for each subscriber and alarm group pair instead. However, on entering an alarm region the safe period to all relevant alarms within the alarm group also needs to be computed.

Despite this additional evaluation step, the number of safe period computations may be considerably reduced by grouping alarms according to spatial locality. Instead of six safe period computations required by the basic safe period technique, only three safe period computations need to be performed as all three alarm groups, $\{AG_i \mid 1 \leq i \leq 3\}$, contain alarms relevant to user 1. Further safe period computations will be performed depending on the number of relevant alarms within the users' current alarm monitoring region. Even though this approach reduces the number of safe period computations it requires considerable additional processing to determine the set of relevant alarm groups for each subscriber and the set of relevant alarms for each subscriber within an alarm group. The lack of subscriber-specificity in the underlying data structure, R-Tree, leads to retrieval of large number of unnecessary alarms. The main cost of this alarm grouping technique is due to these unnecessary alarm checks. This technique proves to be efficient for large number of public alarms as the effect of subscriber-specificity is reduced in the presence of large number of public alarms.

### 3.4.2 Subscriber-Specific Spatial Locality-based Grouping

In contrast to spatial locality-based grouping, subscriber-specific spatial locality-based grouping performs a two level grouping: the first level grouping is on all subscribers and the second level grouping is on spatial alarms relevant to each subscriber. We use a B-tree based implementation to speed up search on subscribers and an R-Tree implementation to capture spatial locality of alarms for each subscriber in order to speed up alarm search. The underlying data structure is a hybrid structure which uses a B-tree for subscriber specific search at the first level and an R-tree for subscriber specific spatial alarm search at the second level. Figure 16(c) shows an example of this grouping. Alarms installed by user 1 are grouped together in $AG_1$ and $AG_4$ and may be fired only when the user is entering the $MBRs$ of $AG_1$ or $AG_4$. Subscriber specific spatial locality-based grouping has two advantages over the basic safe period alarm evaluation and spatial locality based alarm grouping.

First, the number of safe period computations is significantly reduced. Second, each alarm group contains alarms relevant to a single user, thus no irrelevant processing is performed.

Our experimental results show that this approach is efficient in the presence of large number of subscribers and for large number of private and shared alarms. However, this approach is less efficient in presence of large number of public alarms or large number of subscribers with each subscriber subscribed to a very small number of alarms.

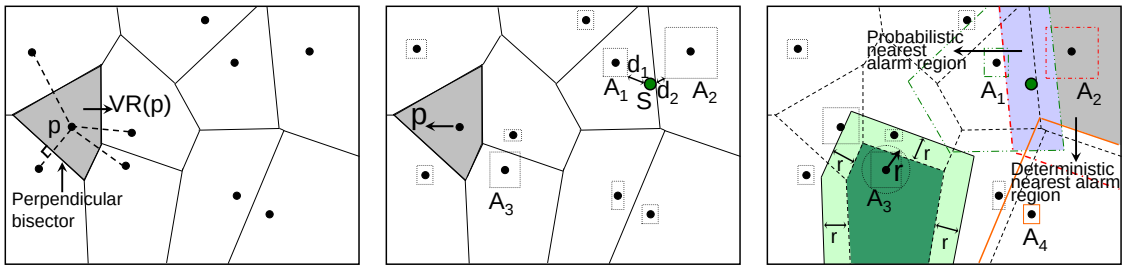
### 3.4.3 Nearest Alarms-based Grouping

Nearest alarms-based grouping allows the system to perform one or only a few alarm checks dependent on the current subscriber position. The idea is to have each location on the map associated with the nearest spatial alarm region(s). In order to perform nearest alarms-based grouping we use an extension of the well known Voronoi diagram geometric structure [21]. The Voronoi diagram for a given set of points *P* in *d*-dimensional space $\mathbb{R}^d$ partitions the space into regions where each region includes all points with a common closest point $\epsilon$ *P*. The common closest point is defined according to some distance metric *dist*. The *Voronoi region VR(p)* corresponding to any point *p* $\epsilon$ *P* contains all points $p_i \epsilon \mathbb{R}^d$ such that,

$$\forall p' \epsilon P, p' \neq p, dist(p_i, p) \leq dist(p_i, p') \tag{12}$$

Figure 17(a) shows the Voronoi diagram for a set of points in two-dimensional space $\mathbb{R}^2$ with euclidean distance metric. The shaded area marks out the *Voronoi region VR(p)* for the point *p*, each point $\epsilon$ *P* is referred to as a *Voronoi site*. Each edge of *VR(p)* is a segment of the perpendicular bisector of the line segment connecting *p* to another point in *P*.

In order to create a Voronoi diagram for spatial alarms we first represent each spatial alarm region *R* by its center point $(x_{cr}, y_{cr})$ and *l*, *h* representing the length and height of the alarm region. We consider the center point of each alarm region as a Voronoi site and create the Voronoi diagram as shown in Figure 17(b). However, this Voronoi structure exhibits two problems. Consider alarm $A_3$ in Figure 17(b). The alarm region overlaps with




(a) Voronoi Diagram (b) Voronoi Extension for Spatial Alarms (c) Overlapping and Non-overlapping Voronoi Regions

**Figure 17:** Nearest Alarms-based Grouping

two adjacent Voronoi regions. Second, consider the subscriber *S* in the figure residing in the Voronoi region of alarm $A_1$. $S$ is at a minimum Euclidean distance $d_1$ from the alarm region of $A_1$ and at a minimum Euclidean distance $d_2$ to the alarm region of $A_2$. Even though $d_2 < d_1$, $S$ may incorrectly identify $A_1$ as the nearest alarm on the basis of the underlying Voronoi diagram. In order to rectify this problem, we introduce an extension to the original Voronoi diagram by extending the boundary of each Voronoi region by the *extension radius* $r$ associated with each point *p* where $r = \sqrt{(\frac{l}{2})^2 + (\frac{h}{2})^2}$. *l, h* denote the length and height of the alarm region associated with center point *p*. The extended Voronoi regions for alarms $A_1$, $A_2$, $A_3$ and $A_4$ are shown in Figure 17(c). Extending the Voronoi region boundaries leads to overlaps among neighboring Voronoi regions, subscribers inside overlapping regions may have more than one possible nearest alarm whereas subscribers inside non-overlapping regions can have only one nearest alarm. We refer to the overlapping regions as *probabilistic nearest alarm region* and the non-overlapping regions as *deterministic nearest alarm region*.

Nearest alarm grouping is efficient for spatial alarm systems that have infrequent addition or removal of alarms and have no hotspots. However, it fails when there is a frequent addition and removal of spatial alarms, since Voronoi diagrams need to be reconstructed each time an alarm is added or removed. In addition, high density of alarms in some areas may also lead to large overlaps among Voronoi regions, reducing the efficiency of the nearest alarm grouping technique.

### 3.4.4 Analytical Model for Safe Period Computation

In this section, we provide an analytical model for estimating the safe period computation cost for the basic safe period approach (BSP), spatial locality-based approach (SLSP) and the subscriber-specific spatial locality-based (SSSL) approach. As mentioned previously, SLSP approach uses an R-Tree structure to perform alarm grouping whereas the SSSL approach uses a two level tree structure. The number of alarm regions allowed in a single alarm group, which translates to the *fan-out* and *fill factor* of a leaf node of the R-Tree, needs to be determined in order to minimize the number of safe period computations. We develop a model which allows us to estimate the appropriate fan-out of a leaf node in order to minimize the safe period computation cost. This result is used to compare the performance of the grouping approaches with the BSP approach.

We consider a workspace with $N$ spatial alarms installed in the system and $n$ users each having an average of $m$ relevant alarms. However, note that $N \neq mn$ as shared and public alarms will be relevant to more than one user. We assume that fraction of private alarms is $p$, fraction of shared alarms is $s$, with each shared alarm relevant to $x$ users on an average, and rest of the alarms are public alarms relevant to all users.

The BSP computation calculates the euclidean distance from the current user position to each relevant alarm. We can use the assumptions stated above to estimate the average number of safe period computations $N_{bsp}$ performed at each evaluation step which is equal to average number of alarms relevant to a single user $N_A$ as,

$$N_{bsp} = N_A = N \cdot \left\{ \frac{p}{m} + \frac{s \cdot x}{m} + (1 - p - s) \right\} \tag{13}$$

SSSL approach distributes the alarms relevant to a single user into multiple groups, where each group only contains alarms relevant to this user. This approach then estimates the euclidean distance to each alarm group, followed by safe period computation for each alarm within the nearest alarm group once the subscriber enters the MBR of the corresponding leaf node. Let $b_{br}$ and $ff$ denote the fan-out and fill factor for a leaf node of the lower level

R-Tree which implies each leaf node can contain $b_{br} \cdot ff$ spatial alarm regions on average. Hence we have $\frac{N_A}{b_{br} \cdot ff}$ alarm groups with all alarms within each group relevant to a single user. The number of safe period computations performed at each evaluation step using the SSSL approach is estimated as,

$$N_{sssl} = \frac{N_A}{b_{br} \cdot ff} + b_{br} \cdot ff \tag{14}$$

We assume that the fill factor $ff$ is set to a constant value which is around 0.7 for best performance of the R-Tree structure [55]. In order to minimize the safe period computations, we can determine $b_{br}$ by setting the first order derivative of $N_{sssl}$ to zero. Hence, we have

$$\frac{d\left(N_{sssl}\right)}{db_{br}} = -\frac{N_A}{b_{br}^2 \cdot ff} + \cdot ff = 0, \tag{15}$$

which implies $b_{br}$ = $\frac{\sqrt{N_A}}{ff}$. Using this value for $b_{br}$, we get,

$$N_{sssl} = \frac{N_A}{\frac{\sqrt{N_A}}{ff} \cdot ff} + \frac{\sqrt{N_A}}{ff} \cdot ff = 2 \cdot \sqrt{N_A} = 2 \cdot \sqrt{N_{bsp}} \tag{16}$$

A similar analysis of the SLSP approach shows that, $N_{slsp} \geq 2 \cdot \sqrt{N_{bsp}}$. However, as this approach mixes alarms from different users in an alarm group the worst case number of safe period computations can be $N_{sslp} = N_{bsp} + 1$. This situation may arise when all $N_A$ alarms relevant to a user are distributed across $N_A$ different groups which will require $N_A$ safe period computations for the alarm groups and a single safe period computation for the relevant alarm within that group.

## *3.5 Subscriber Mobility-based Optimizations*

In order to determine the minimum safe period for any subscriber it is essential to compute the safe periods to all relevant alarms or alarm groups using the approaches described in Section 3.4.1 and 3.4.2. We have discussed three different alarm grouping techniques, which improve the basic safe period alarm evaluation by promoting two-phase safe period computations: First, safe period computation is only performed for each subscriber and

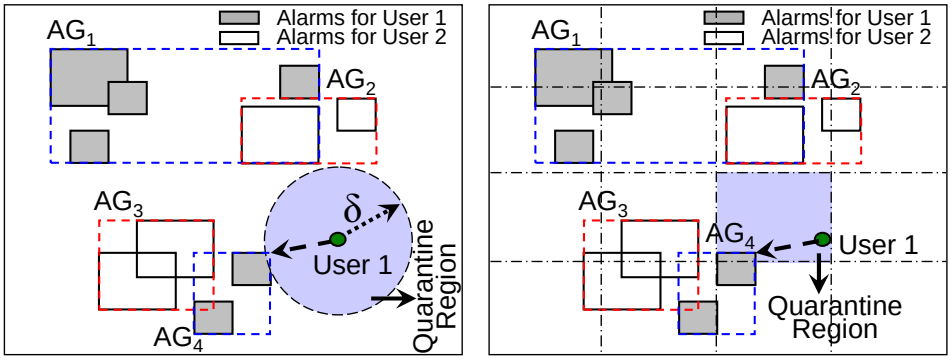
alarm group pair, and upon entering the alarm group monitoring region, we switch to the safe period computation for each subscriber-alarm pair.

In this section, we introduce subscriber mobility-based optimizations, which further reduce the amount of unnecessary safe period computations. The main idea behind the subscriber mobility-based optimization is to avoid safe period computations for those spatial alarms that are far away from the current location of the mobile subscriber. We observe that computing the safe periods for all spatial alarms regardless of how far away they are from the current position of their subscribers can lead to wasteful computation, since for each mobile subscriber, alarms at remote distances will never be fired before the expiration of the safe periods of nearby alarms. Our experiments show that *subscriber mobility-based safe period optimization* is highly effective for improving performance and scalability of spatial alarm processing systems.

One way to implement subscriber mobility-based optimization is to define a moving spatial area around each mobile user which serves as the *quarantine region*. We use a system-defined *quarantine region* to set the alarm monitoring area for each mobile user and allow different sizes of the quarantine region based on a number of factors, such as the velocity of the mobile subscriber, the alarm density near the mobile subscriber and the number of alarms installed per subscriber. For each subscriber, at any given time instant safe periods are computed only for relevant alarm groups (or alarms) whose alarm group MBRs (or alarm regions) overlap with this moving quarantine region. Clearly by focusing on computing safe periods and performing safe period alarm evaluation only for alarm groups (or alarms) near each mobile subscriber, this optimization can effectively reduce the complexity of safe period computations and enhance the system scalability.

We describe two different methods for determining the appropriate quarantine region for each subscriber and discuss the pros and cons of each when applying mobility-based

optimization to our alarm grouping mechanisms. The first method is the *range-based subscriber mobility optimization*, which uses a system-supplied radius $\gamma$ to determine the quarantine region for each subscriber. The second method uses a pre-defined grid and defines the grid cell in which the mobile subscriber currently resides as the quarantine region. When the mobile user moves to a new grid cell, her quarantine region changes; thus the set of alarms to be monitored changes. We call this method *grid-based subscriber mobility optimization*. Figure 18 shows examples for these two methods.




(a) Range-based optimization (b) Grid-based optimization

**Figure 18:** Subscriber Mobility-based Optimizations

For range-based mobility optimization, we currently use a system-supplied $\gamma$ as the default quarantine radius. Given a mobile subscriber, we first need to identify the set of alarms subscribed by her and then determine which of her alarm groups are intersecting with the quarantine region defined by the given quarantine radius. We compute the safe period only for those alarm groups whose alarm monitoring regions overlap with the quarantine region (see Figure 18(a)). The retrieval of relevant alarms whose alarm regions intersect with the quarantine region can be done by using the existing R-tree index for spatial locality-based grouping or the two level B-tree plus R-tree index for subscriber-specific spatial locality-based grouping.

For grid-based mobility optimization, the same principles are applied. The only difference lies in the mechanism used to determine the quarantine region. The grid-based optimization is designed to incorporate subscriber mobility optimization with the nearest

alarm grouping. Concretely, the quarantine region is defined using a grid overlaid on top of the Voronoi diagram discussed in Section 3.4.3. The size of the quarantine region depends on the size of the grid cell $\alpha \times \beta$, which are system defined parameters. For all Voronoi regions that overlap with the quarantine region of a mobile subscriber, the system retrieves all nearest alarms for each Voronoi region and performs safe period computations for these alarms. The set of relevant alarms and their safe periods need to be recomputed when the subscriber moves out of the current cell and enters another cell. The grid overlay will also help in mitigating the costs associated with storing and querying complex shaped Voronoi regions.

## *3.6 Experimental Evaluation*

In this section, we report our experimental evaluation results. We show that our safe period-based framework and optimization techniques for spatial alarm processing are scalable and effective while maintaining high accuracy.

### 3.6.1 Experimental Setup

Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS [8]) in Spatial Data Transfer Format (SDTS [10]). A transport layer of 1:24K Digital Line Graphs (DLGs) is used to extract the road-based network and the data is converted to the Scalable Vector Graphic (SVG [9]) format using the GlobalMapper tool [5]. The simulator extracts the road network information for three different road classes – *expressway*, *arterial* and *collector* roads. Traffic volume data from [53] is used to estimate the number of vehicles for different road classes; vehicles are randomly placed on the road network according to the traffic densities. The simulator simulates the motion of vehicles on roads with appropriate velocity information based on road classes; at intersections, vehicles may move in any direction with attached probability values. We use a map from Atlanta and surrounding regions of Georgia, which covers an area larger than 1000 $km^2$, to generate the

**Table 2:** Motion Parameters

| **Road type (speeds)** | **Expressway** | **Arterial** | **Collector** |
|---|---|---|---|
| **Mean (km/h)** | 90 | 60 | 50 |
| **Std. dev. (km/h)** | 20 | 15 | 10 |
| **Traffic data (cars/h)** | 2916.6 | 916.6 | 250 |

trace. Our experiments use traces generated by simulating vehicle movement for a period of fifteen minutes, results are averaged over a number of such traces. Results for longer time periods show similar patterns. Table 2 lists mean speeds, standard deviation and traffic volume values for each road type. Default traffic volume values allow us to simulate the movement of a set of 20,000 vehicles on the road network for the map. Each vehicle generates a set of position parameters during the simulation which are evaluated against the generated spatial alarm information. The default spatial alarm information consists of a set of 10,000 spatial alarms installed uniformly over the entire map region; with default settings, around 65% of the alarms are private, 33% shared and the rest are public alarms. This simulator setup allows us to the test the robustness of our framework under realistic mobility patterns.

### 3.6.2 Performance Metrics

We identify the following requirements for efficient server-centric processing of spatial alarms based on which we identify the performance metrics for evaluating system performance.

**Success Rate:** The success rate achieved by the spatial alarm service provider is the most critical performance evaluation criterion. The service *must not* miss any of the spatial alarms as it is essential to deliver all alarms at the appropriate time in order to maintain the Quality of Service (QoS) desired by the user. Note that successful alarm evaluation does not necessarily translate to successful alarm delivery as alarm delivery may be affected by other factors like intermittent connectivity. We consider success rate only for alarm evaluation at the server side.

**Scalability:** The system architecture should be able to support large number of clients with large number of installed alarms while guaranteeing the QoS requirements which are very stringent in this scenario. Intelligent optimization techniques deployed at the server ease the processing load which in turn aids the scalability of the system.

Based on the above mentioned requirements we identify the following performance metrics for measuring the efficiency of our approach:

*Number of alarm evaluation steps:* Each alarm evaluation step involves evaluating current user position information against relevant spatial alarms. Safe period optimizations aim at reducing the number of evaluation steps by determining a safe period during which the evaluation steps can be avoided.

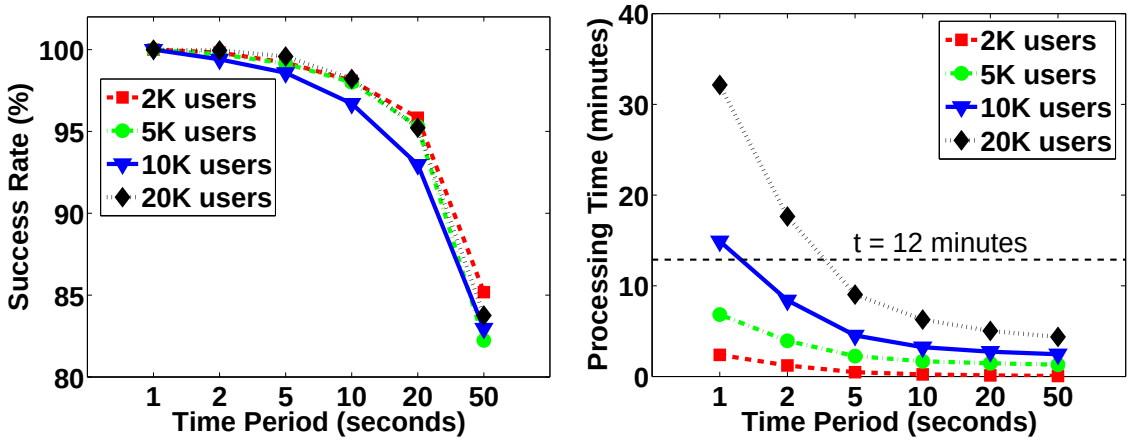
*Number of safe period computations:* Safe period computation involves calculating the minimum time period for a client before it needs to perform spatial alarm evaluation. This metric allows us to measure the advantage gained by our grouping techniques compared to basic safe period computations.

*Success Rate:* We must ensure that our optimizations do not affect the correct functioning of the spatial alarm processor. Ideally, we want zero alarm misses for alarm processing by the system.

*Evaluation time:* The evaluation time determines the ability of the system to handle the processing load. It consists of two main components: *alarm evaluation time* and *safe period computation time*. Evaluation time is measure over the entire simulation period averaged over a large number of runs.

### 3.6.3 Experimental Results

We evaluate the safe period-based approach to spatial alarm processing through four sets of experiments.The first set of experiments measures the performance of periodic alarm evaluation by varying the time period and measuring success rate and processing time. We show that the periodic approach does not scale.The second set of experiments compares the




(a) Success Rate

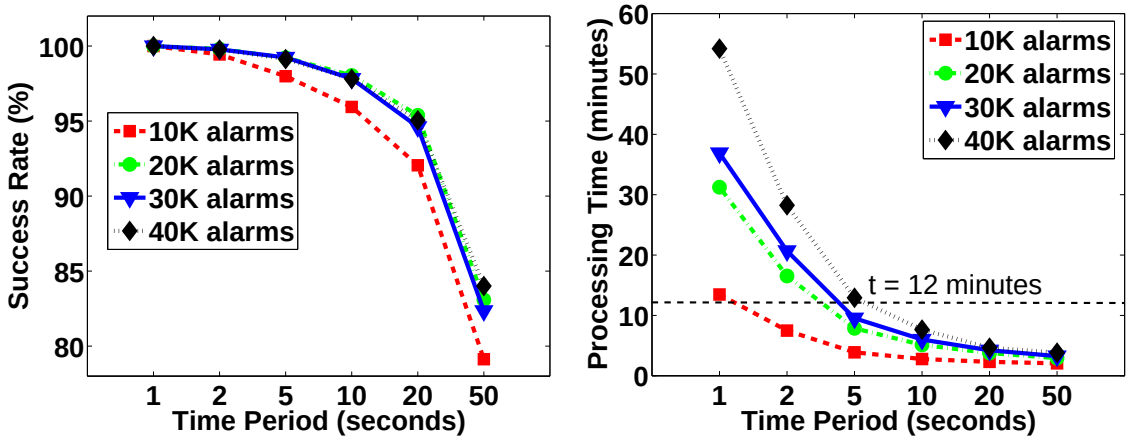
(b) Alarm Processing Time

**Figure 19:** Scalability with Varying Number of Users

basic safe period alarm evaluation against the periodic approach, and shows that the safe period based alarm processing offers much higher success rate with lower alarm evaluation time. The third set of experiments illustrates the effect of varying *quarantine radius* on the range-based subscriber-mobility optimization technique. The final set of experiments compares the performance of the various grouping-based and mobility-based safe period optimizations against the basic safe period approach exhibiting the scalability of our optimizations. We show that the mobility-based optimizations outperform all other techniques in terms of the number of safe period computations.

#### *3.6.3.1 Scalability Problems of Periodic Alarm Evaluation Technique*

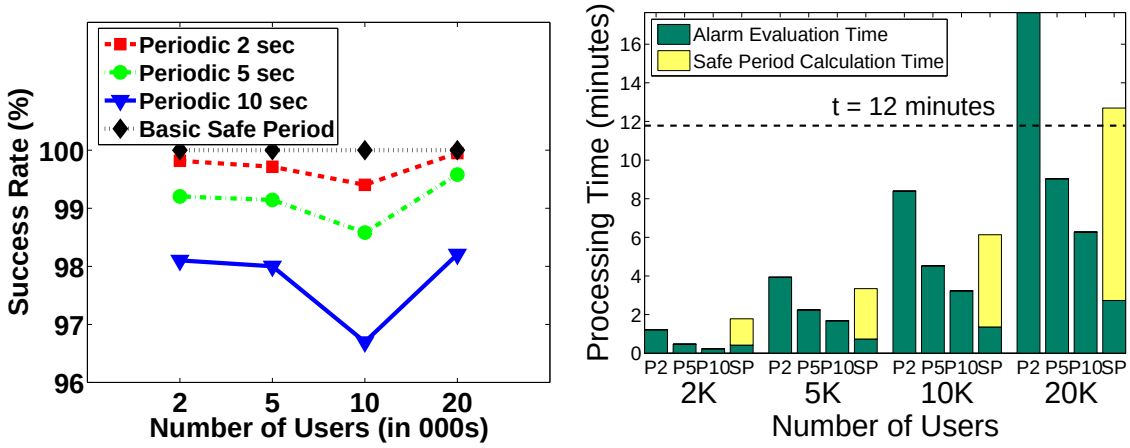
In this first set of experiments, we measure the scalability of the periodic alarm evaluation technique with varying number of users and varying number of alarms. Figure 19 displays the results as we vary the number of users from 2K to 20K. The time period $t_p$ for periodic alarm evaluation is varied from 1 second to 50 seconds. As can be seen from Figure 19(a), the success rate for alarm evaluation is 100% only if $t_p$= 1 second; for higher $t_p$ success rate starts falling, even with $t_p$= 2 seconds the success rate does fall to 99.9% which may not be acceptable from QoS viewpoint as this translates to a significant number of alarm misses. The sequence of alarms to be triggered for 100% success rate are determined from




(a) Success Rate (b) Alarm Processing Time

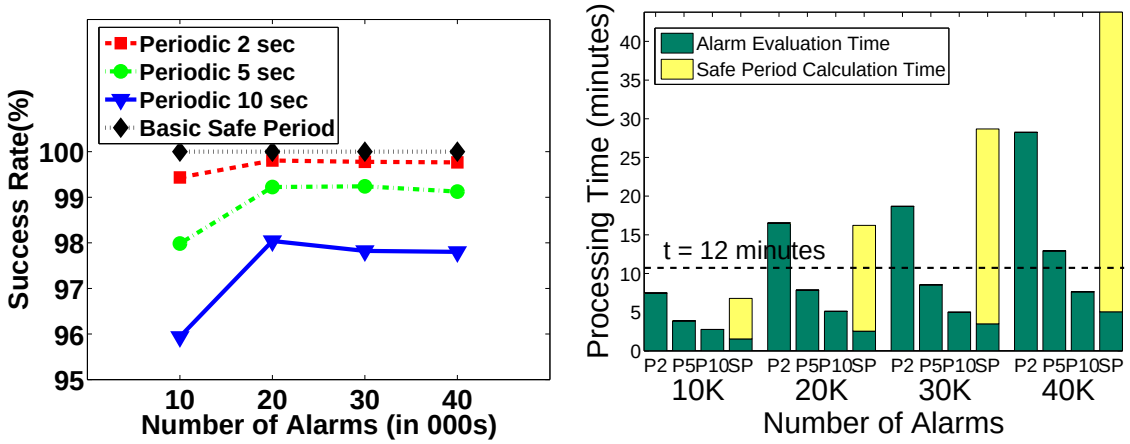
**Figure 20:** Scalability with Varying Number of Alarms

a trace generated with extremely frequent location update information for each user in the system. For $t_p$= 50 seconds the success rate falls to 81-82%. Similar drop in success rate is experienced in all cases as we vary the number of users in the system from 2K to 20K. The alarm processing time is plotted in Figure 19(b). Our traces are of fifteen minutes duration; considering that the system may be able to spend around 80% of this time for processing spatial alarms we set the maximum processing time available to the system at $t$=12 minutes as indicated by the horizontal dotted line in Figure 19(b). As can be seen from the figure, for 10K users the system is unable to process alarms at $t_p$=1 seconds, thus failing to attain 100% success rate. For 20K users, this scalability problem becomes worse and the system is able to evaluate alarms only at $t_p$=5 seconds. Figure 20 shows the results for a set of 10K users as we vary the number of alarms from 10K to 40K. The success rate, as shown in Figure 20(a), again exhibits a similar drop on increasing $t_p$. The alarm processing time shown in Figure 20(b) displays the inability of the system to scale to large number of alarms. From these results, we conclude that periodic evaluation technique is unable to scale to a large number of users and large number of alarms.




(a) Success Rate (b) Alarm Processing Time

**Figure 21:** Basic Safe Period Optimization with Varying Number of Users




(a) Success Rate (b) Alarm Processing Time

**Figure 22:** Basic Safe Period Optimization with Varying Number of Alarms

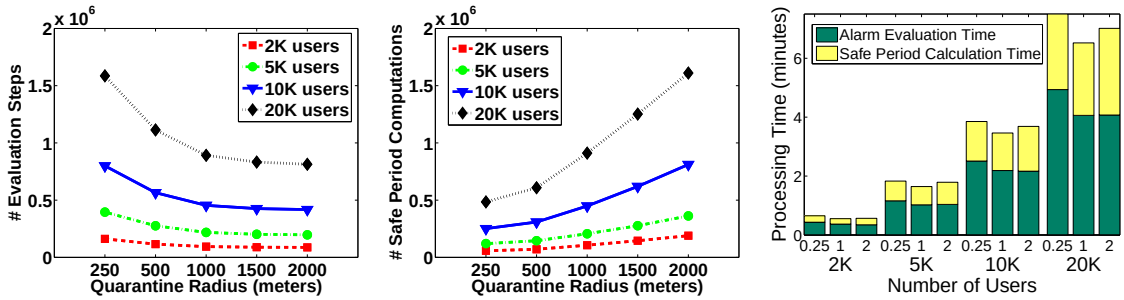
#### *3.6.3.2 Performance Comparison with Basic Safe Period Approach*

In this section, we compare the performance of basic safe period approach against the periodic evaluation technique to display that basic safe period optimizations reduce alarm evaluation time considerably but excessive amount of safe period computations affect the scalability of the system. We display the results for periodic approach with $t_p$=2 second, $t_p$= 5 seconds, $t_p$=10 seconds and the basic safe period optimization as discussed in Section 3.3 (P2, P5, P10 and SP in Figures 21(b) and 22(b) respectively). Figure 21 displays the success rate and processing time as we vary the number of users from 2K to 20K. Figure 21(a)

displays that the success rate is 100% for basic safe period approach and all periodic approaches miss at least a few alarm triggers. Figure 21(b) displays the alarm processing time for P2, P5, P10 and SP with varying number of users. The safe period approach has much lower alarm evaluation time compared to periodic approaches and the figure displays that it is *almost* scalable to 20K users with 100% success rate. Despite the low alarm evaluation time, the approach requires significant amount of safe period computations and has high processing time as can be seen from the figure. Figure 22 displays the results for a set of 10K users with varying number of alarms as we increase the number of alarms from 10K to 40K. The success rate, shown in Figure 22(a), again displays similar patterns as with varying number of users. The alarm processing time, as shown in Figure 22(b), displays the inability of our basic safe period approach to scale to large number of alarms. In presence of even 20K installed alarms, the approach has excessive safe period computation time which pushes the overall processing time beyond the 12 minute limit determined earlier. Our alarm grouping and subscriber mobility-based techniques provide optimizations to overcome this problem as displayed by the experimental evaluation in Section 3.6.3.4.

#### *3.6.3.3 Internal System Parameters*

This set of experiments determines the appropriate parameters for the quarantine radius $\gamma$ for range-based subscriber mobility optimization. Figure 23 displays the results obtained for the number of alarm evaluation steps, number of safe period computations and overall processing time with varying values for $\gamma$. We vary the radius from 250m to 2000m and observe the above parameters. The number of users is varied from 2K to 20K to observe results across a wide range of number of users in the system. As can be seen from Figure 23(a) the number of alarm evaluation steps steadily decreases as we increase $\gamma$. Smaller $\gamma$ values will calculate lower safe periods in absence of any alarms (or alarm groups) within the quarantine region. Hence, for lower values of $\gamma$ the safe period computations are more conservative. This trend is common as we vary the number of users from 2K to 20K. The



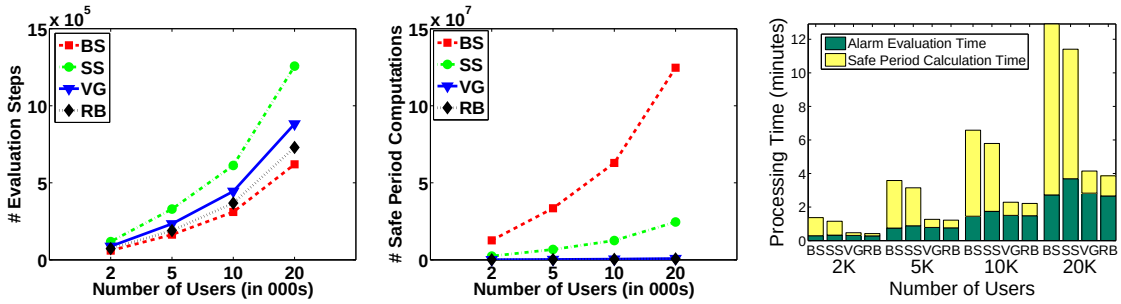

(a) # Alarm Evaluation Steps (b) # Safe Period Computations (c) Alarm Processing Time

**Figure 23:** Results with Varying Quarantine Radius Values

number of safe period computations steadily rises as we increase $\gamma$ (Figure 23(b)). This trend is also as expected because larger $\gamma$ implies more alarms (or alarm groups) will lie within the quarantine region. A trade-off between these two factors is required to determine the appropriate value for $\gamma$. Figure 23(c) displays the overall processing time for different $\gamma$ values across varying number of users. The figure displays the alarm evaluation time and safe period calculation time for $\gamma \epsilon \{0.25km, 1km, 2km\}$ for each set of users; results for other $\gamma$ values are omitted from this graph to avoid clutter. As we vary the number of users, from 2K to 20K, we observe that for each set of users the overall processing time is minimum for $\gamma$=1000m. We choose this as the appropriate value for $\gamma$ for further experimentation.

#### *3.6.3.4 Scalability of Safe Period Evaluation Techniques*

We now discuss the performance of the safe period optimization techniques to test the scalability of our framework. Figure 24 shows the number of alarm evaluation steps, number of safe period computations and the alarm processing time required by each approach-Basic Safe Period Optimization (BS), Subscriber-Specific Spatial Locality (SS), Voronoi Grid-Based (VG) and the Range-based Subscriber Mobility Optimization (RB). Results for Spatial Locality-based grouping show expected trends but this approach has high overall processing time as the system needs to perform significant amount of computation to determine relevance of alarms/alarm groups for each subscriber (see Section 3.4.1). Hence,

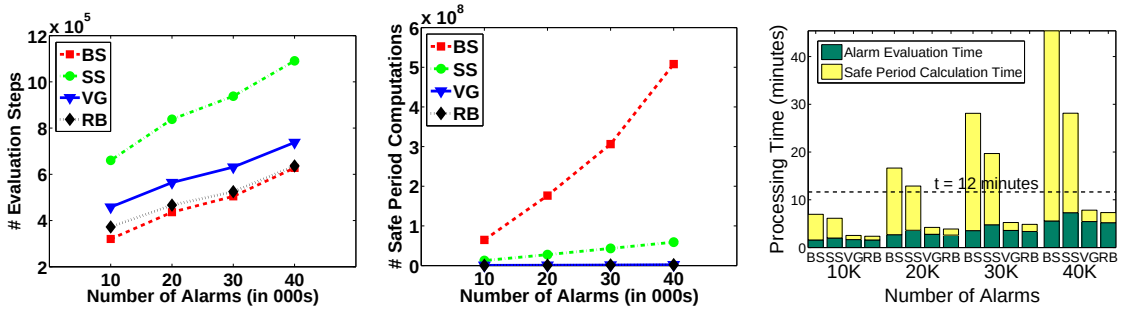



(a) # Alarm Evaluation Steps (b) # Safe Period Computations (c) Alarm Processing Time

**Figure 24:** Safe Period Optimizations with Varying Number of Users

we exclude this approach from the results below.

Figure 24(a) displays the number of alarm evaluation steps required by each approach. Basic safe period measures the safe period to each relevant alarm and uses this safe period to avoid further evaluations. As a result, this approach has to perform a low number of alarm evaluations but each evaluation step will involve a very large number of safe period computations. Hence the number of safe period computations for this approach is extremely large (Figure 24(b)) which makes this approach overall computationally expensive as can be seen from the total alarm processing times in Figure 24(c). Subscriber-specific spatial locality grouping incurs a large number of alarm evaluation steps as can be seen from Figure 24(a). This approach first evaluates safe period for each alarm group; once the user enters an alarm monitoring region another evaluation step is required to determine the safe period to all alarms lying within the alarm monitoring region. Further, the algorithm needs to keep a check on its position inside the alarm monitoring region and switch to per alarm group-based safe period computations, once subscriber moves outside the current alarm monitoring region. These additional evaluation steps imply that this approach will incur a larger number of alarm evaluation steps with each evaluation step requiring a small number of safe period computations: either for each alarm group or for all alarms lying within the current alarm monitoring region. Thus the number of safe period computations required by this approach is much lower than the basic approach despite the larger number of alarm evaluation steps. Consequently, the overall processing time for SS is lower than




(a) # Alarm Evaluation Steps (b) # Safe Period Computations (c) Alarm Processing Time

**Figure 25:** Safe Period Optimizations with Varying Number of Alarms

the BS approach as can be seen from Figure 24(c).

The VG and RB approaches lower the number of alarm evaluation steps by considering only alarms or alarm groups within the quarantine region. In this set of experiments the quarantine radius $\gamma$ for RB is set to 1000m as determined by the results in the previous section. Similarly, VG grid cell size is set to 1000m $\times$ 1000m. The number of evaluation steps for these approaches is still larger than the number of evaluation steps used by the basic approach as the safe periods computed by this approach may be lower than the safe period computed by the basic approach, in case no relevant alarms/alarm groups lie within the quarantine radius range or the current grid cell of the subscriber. In absence of any alarms/alarm groups within the quarantine region, the safe period for these approaches is calculated as the time required by user to reach the edge of the quarantine region.

However, each alarm evaluation step involves a very small number of safe period calculations leading to an extremely small number of safe period computations (in Figure 24(b) results for VG and RB are almost overlapping). Consequently, the overall processing times for these two approaches are significantly lower than other approaches. Figure 25 displays the results for a set of 10K users as we vary the number of alarms from 10K to 40K. These results confirm that our mobility-based optimizations can scale to a large number of alarms. As shown in Figure 25(c), even for 40K alarms VG and RB approaches have a processing time lower than the 12 minute limit determined earlier. From these results we can conclude that our safe period optimizations significantly aid the scalability of the system.

## *3.7 Related Work*

An event-based location reminder system has been advocated by many human computer interaction projects [78, 97, 40, 80, 66]. Understandably, the primary focus of the work is from the point of view of the usability of such systems. Some of the work provides extensive user evaluation studies which establish the usefulness of location-based reminder systems beyond doubt. However, none of these approaches deal with the system oriented issues which need to be resolved to make such systems feasible.

In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand. User-defined triggers can be initiated when new relevant information which is of personal interest to the user is detected by the system [76, 30]. In addition to monitoring continuously changing user information needs, spatial alarm processing systems also need to deal with the complexity of monitoring user location data in order to trigger relevant alerts in a non-intrusive manner.

Applications like Geominder [4] and Naggie [6] already exist which provide useful location reminder services using cell tower ID and GPS technology, respectively. Client-based solutions for spatial alarm processing should focus on efficiently evaluating spatial alarms while preserving client energy. Our server-centric architecture makes it possible for users to share alarms and make use of external location information monitoring services which provide relevant location-based alerts. A server-centric approach is also essential for extending the technology to clients using cheap location detection devices which may not possess significant computational power. Even for clients with significant computing resources, energy and bandwidth consumption remain major bottlenecks and numerous works have dealt with the problem of energy conservation in mobile devices [44, 45, 86]. In this work, we propose a scalable and efficient centralized architecture for processing spatial alarms to resolve the above issues.

# CHAPTER IV

# SAFE REGION TECHNIQUES FOR FAST SPATIAL ALARM EVALUATION

## *4.1 Introduction*

Location provides one of the most powerful tools for personalization of mobile services. The ability of GPS assisted devices to provide accurate location information with a reasonably high degree of temporal precision makes it feasible to use a multitude of location-based applications which constantly survey user surroundings to deliver useful location-based information on demand. Cheaper GPS devices integrated with mobile phones are further pushing the demand for location-based services with development of mobile platforms like Android [11] and iPhone SDK [12]. The market for GPS devices is growing at around 20% annually with 90% of the devices being portable navigation devices. By end of 2012, mobile phones are expected to have around 78% of this market as GPS is incorporated into them [15].

Due to the multitude of available applications, location-based services become dependent on personal preferences and interests of the users. Spatial alarms [28, 87] provide a useful abstraction for modeling a large number of personalized location-based applications. Consider the following examples:

**Example 1: Location-based advertisements.** Macy's store may set a spatial alarm around its store locations in Atlanta for *"sending e-coupons for a 20% discount to all gold members within a five mile radius of it store locations"*. This allows the store to limit delivery of coupons to customers in the vicinity of the store. Customers may choose to subscribe to spatial alarms installed on Macy's stores or otherwise, thus, personalizing delivery of information at their end.

**Example 2: Location-based reminders [97].** A user may set a spatial alarm on her favorite grocery store for *"reminding her to buy groceries whenever she is one mile away from the store over the weekend"*. The installation of such an alarm allows the user to offload an essential function of remembering to buy groceries to her mobile phone!

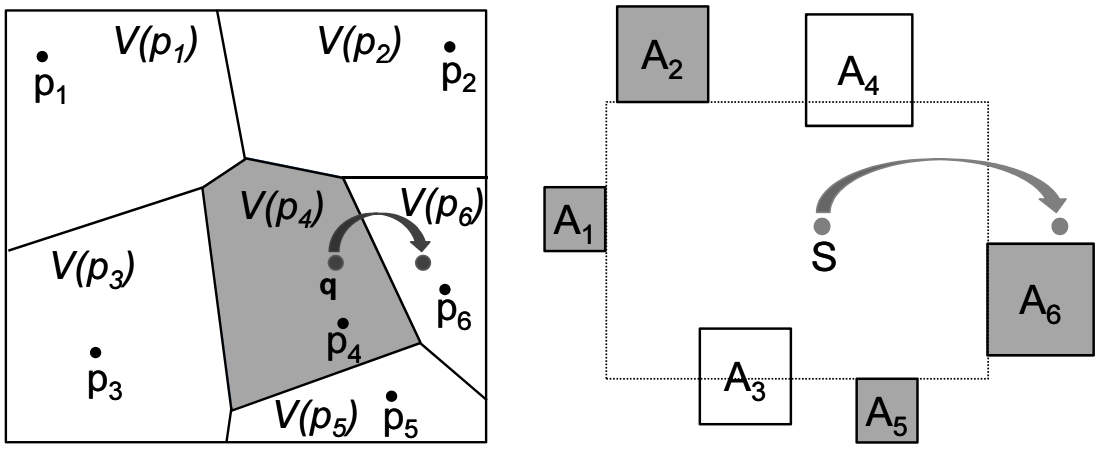
**Example 3: Location-enhanced social networking.** The concept of location when integrated into social networking applications like Facebook [13] allows the ability to add physical presence-based functionality. For example, a user may install an alarm on all friends which *"informs her whenever any of her friends are within a two mile vicinity of her current location during office lunch hours"*. As this requires monitoring of friends' location information, dealing with location privacy issues is also essential for such alarms [27].

Client-centric [87] and server-centric [28] architectures for spatial alarm processing are both feasible. A client-based architecture is limited in application, due to absence of interaction among users and information delivery systems, but attractive due to it simplicity. On the other hand, from the user perspective, a server-centric architecture has various advantages not limited to offloading of alarm processing to a centralized server, sharing of alarms among users, ability to subscribe to alarms installed by large number of information delivery systems. However, centralization of alarm processing leads to the alarm processing server becoming a bottleneck as the server needs to process large number of alarms installed by a large number of mobile users.

We consider a spatial alarm processing system which hosts spatial alarms installed by a large number of mobile users and information monitoring systems. Users subscribe to these alarms and report their position to the server; the server receives user position information and processes this information against the installed alarms triggering relevant alarm notifications. Processing of spatial alarms requires meeting two demanding objectives: high accuracy, which ensures no alarms are missed, and high scalability, which guarantees that alarm processing is highly efficient and scales to large number of spatial alarms and growing base of mobile users. The conventional approach to similar problems involves periodic

evaluations at a high frequency. Each spatial alarm evaluation can be conducted by testing whether the user is entering the spatial region of the alarm. High frequency is essential to ensure that none of the alarms are missed. Though periodic evaluation is simple, it can be extremely inefficient due to frequent alarm evaluation and the high rate of irrelevant evaluations. This is especially true when the mobile user is traveling in a location that is distant from all her location triggers, or when all her alarms are set on spatial regions that are far apart from one another. Safe period-based approaches [28] allow the system to overcome these deficiencies by adaptively computing a safe period for each user; no alarm processing needs to be performed for the user before the expiry of its safe period. However, these approaches heavily rely on future motion estimation of the user for computing the safe period. Pessimistic motion estimations may still lead to frequent alarm evaluation leading to excessive load on the alarm processing server. Optimistic motion estimations can lead to missed alarms which defies the high accuracy criterion.

In this work, we present *safe region*-based approaches for spatial alarm processing. We show that the safe region approach, when applied to our alarm processing problem, enables *distribution of the processing load between the server and the large number of mobile users* significantly aiding server scalability at the cost of nominal energy consumption at the client end. The *Voronoi Diagram* [21], as shown in Figure 26 is a classical example of safe region-based techniques. Consider the set of points, as shown in Figure 26(a), displaying the Voronoi region, $V(p_i)$, for each point $p_i$. Processing a simple *moving nearest neighbor query* for query point $q$ simplifies to identifying the Voronoi region associated with the current location of $q$. In the above example, as long as $q$ is within the region $V(p_4)$, $p_4$ remains the nearest neighbor for $q$. In this scenario, the Voronoi region $V(p_4)$ (shaded region) is a safe region for the moving NN query; no processing of any information is necessary as long as the moving query point lies within this safe region. Once $q$ moves into the Voronoi region $V(p_6)$, the safe region changes and query recomputation is required. Figure 26(b) provides an example of safe region for the spatial alarm processing problem.




(a) Voronoi Safe Region (b) Rectangular Safe Region

**Figure 26:** Examples of Safe Region

Spatial alarm regions $A_1$, $A_2$, $A_5$ and $A_6$, represented by shaded rectangular regions, are relevant to the subscriber $S$, whereas, alarm regions $A_3$ and $A_4$ are not relevant to this subscriber. The safe region is represented by the rectangular region touching the boundaries of the relevant alarm regions. As long as $S$ is located within this safe region, none of the relevant alarms can be triggered; as soon as $S$ moves out of this rectangular region a new safe region needs to be computed.

We make three unique contributions in this work: (i) Firstly, we explore safe region-based techniques for spatial alarm processing. Concretely, we propose the *Maximum Weighted Perimeter Rectangular Safe Region* approach and two approaches for representing rectilinear polygonal shapes using bitmap encoding, namely, *Grid Bitmap Encoded Safe Region* and *Pyramid Bitmap Encoded Safe Region*. Our experimental evaluation shows that the safe region-based approaches outperform periodic evaluation and safe period-based approaches. (ii) Secondly, we consider different heuristics based on size, shape of the safe region and study their affect on the bandwidth, energy and server load resource consumption. (iii) Last but not the least, unlike previous safe region approaches, our framework supports heterogeneous client capabilities using bitmap encoding techniques.

The rest of the chapter is outlined as follows. Section 4.2 introduces basic concepts associated with spatial alarms, safe region and a grid overlay to limit safe region computation

costs. Section 4.3 describes our rectangular safe region approach followed by the bitmap encoding techniques in Section 4.4. A detailed experimental evaluation is conducted in Section 4.5 using a real world road network. Related work is presented in Section 4.6 followed by the conclusion.

## *4.2 Preliminaries*

### 4.2.1 Spatial Alarms

A spatial alarm expresses a location-based information need of a subscriber $s \in S$ around a *location of interest*. The alarm trigger requires that the subscriber be informed as soon as she enters a spatial region $R$ around the location of interest and the non-spatial constraints associated with the trigger are satisfied.

Spatial alarms can be categorized as *private*, *shared* or *public* alarms depending on the scope of subscribership of the alarm. Private alarms are relevant to a single subscriber authorized to install or remove the alarm. A subscriber may install an alarm on the neighborhood grocery store reminding her to purchase groceries when she is within a one mile radius of the store and special discounts are available on her desired items. Shared alarms are installed by a subscriber of the alarm and may be shared with a group of users; for example, in the above scenario the subscriber may wish to share the alarm on the grocery store with other members of her household. Public alarms are relevant to all subscribers in the system; examples of such alarms are warning notifications against hazardous road conditions.

Motion characteristics of the subscriber and the alarm target provide another criterion for alarm categorization. The first class is the *Mobile Subscribers Static Target* alarms, where alarm targets are still objects such as restaurants, hospitals, etc. The second class is referred to as the *Static Subscribers Mobile Target* alarms, where alarm targets are moving but the position of mobile subscribers remains unchanged during the alarm validity period. A typical example of such alarms is *"tell me when the bus is within two miles of the bus stop*

*near my office"*. The third class, where both alarm subscribers and targets are moving, is called the *Mobile Subscribers Mobile Target* alarms. Location enhanced social networking applications typically install such alarms.

### 4.2.2 Safe Region

**Definition 4.2.1** *We define the Safe Region, denoted by $\Psi_s$ for each user $s \in \mathcal{S}$, such that, (i) as long as the user's position lies within this region the probability of the user entering any of its relevant spatial alarm regions is zero. (ii) If the user position lies inside one or more relevant spatial alarm regions, the intersection of the spatial alarm regions forms the safe region for the user. In this case, as long as the client remains inside the safe region the probability of any alarms other than those associated with this safe region being triggered is zero. Consequently, as long as the user $s$ lies within its computed safe region any location updates may be dropped without performing alarm evaluation.*

$$Pr[l_s(t) \Rightarrow \mathcal{A}_i \,|\, l_s(t) \in \Psi_s] = 0 \tag{17}$$

$l_s(t)$ denotes the position information for subscriber $s$ at time $t$, $\mathcal{A}_i$ where $i \in [1...n]$ denotes the set of alarms relevant to subscriber $s$ and $\Rightarrow$ denotes an alarm trigger action. The spatial alarm processing server computes a safe region $\Psi_s$ for each subscriber and communicates this safe region to the subscriber. The subscriber is responsible for monitoring its position within the safe region. Once the subscriber moves out of its safe region it provides a location update to the server which performs alarm processing and recomputes the safe region.

The safe region approach aims at minimization of resource consumption at the server and the client. In the distributed environment we consider here, where client-server communication occurs through a wireless channel, safe region computation must satisfy the following constraints:

**Lightweight Construction.** The safe region computation should induce a low processing

overhead at the server as this computation may need to be performed frequently for a large number of mobile users.

**Compact Representation.** The safe region should have a compact representation as it needs to be communicated back to the user resulting in consumption of downstream bandwidth. For example, a rectangular-shaped safe region requires only two points (bottom-left and top-right) for representation and may be considered to be suitable to meet this requirement.

**Fast Containment Check.** Mobile users need to monitor their position within the safe region almost continuously. A simple containment check will enable the clients to perform this function with low energy consumption.

**Device Heterogeneity.** Their may exist vast diversity in resource capabilities of clients participating in the spatial alarm processing system. Safe region computation techniques which are flexible, adaptive and sensitive to device computational capabilities are suitable to meet this requirement. Our bitmap safe region techniques, introduced in Section 4.4, provide an elegant solution which supports this requirement.

### 4.2.3 Grid Overlay for Safe Region Computation

In its simplest form safe region for any subscriber comprises of the region covered by the entire Universe of Discourse (or map) except the relevant spatial alarm regions. This poses two problems: (i) Firstly, it amounts to communicating information for all relevant alarms to the subscriber which leads to heavy communication costs. (ii) Secondly, it leads to excessive computation costs at the server which needs to consider a large number of alarms for each safe region computation. In order to deal with the computation costs, we overlay a grid on top of the Universe of Discourse which allows us to limit the defined safe region to the vicinity of the current subscriber position.

**Definition 4.2.2** *In our framework, we map the Universe of Discourse* $\mathbb{U} = Rect(x, y, w, h)$ *onto a grid* $\mathbb{G}$ *of cells.* $(x, y)$ *represents the bottom-left corner and* $w, h$ *represent the width*

*and height of* $\mathbb{U}$. *Formally, a grid corresponding to the universe of discourse* $\mathbb{U}$ *can be defined as* $\mathbb{G}(\alpha, \beta) = \{C_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N, C_{i,j} = Rect(x + i \cdot \alpha, y + j \cdot \beta, \alpha, \beta), M = \lceil w/\alpha \rceil, N = \lceil h/\beta \rceil\}$. $C_{i,j}$ *is an* $\alpha \times \beta$ *rectangular area representing the grid cell that is located in the* $i^{th}$ *column and* $j^{th}$ *row of the grid* $\mathbb{G}$.

Considering our above definition of a grid we can define a mapping from any point $\vec{p} = (p_x, p_y)$ in the Universe of Discourse to the Grid, $f : \mathbb{U} \Rightarrow \mathbb{G}$.

**Definition 4.2.3** *Let* $\vec{p} = (p_x, p_y)$ *be any point in the Universe of Discourse* $\mathbb{U}$. *Let* $C_{i,j}$ *denote a cell in the grid* $\mathbb{G}(\alpha, \beta)$. $f(\vec{p})$ *is a position to grid cell mapping, defined as* $f(\vec{p}) = C_{\lceil \frac{p_x - x}{\alpha} \rceil, \lceil \frac{p_y - y}{\beta} \rceil}$ *where* $(x, y)$ *denotes the bottom-left corner of* $\mathbb{U}$.

Our safe region approaches utilize this grid overlay to limit the safe region computation for each subscriber. The grid overlay can be used to limit the safe region computation to an area comprising of the current cell of subscriber $s$. We define the *monitoring region* $\zeta_s$ inside the current grid cell below and proceed to describe our algorithms for safe region computation.

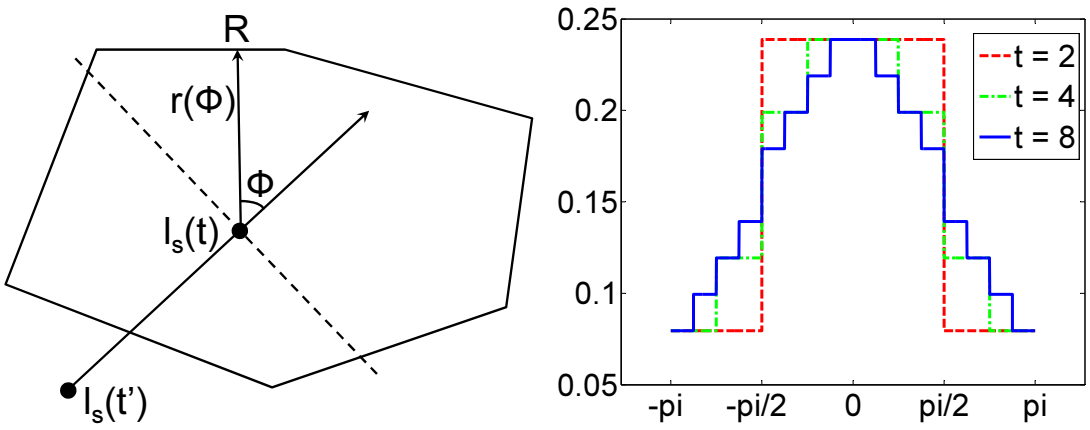
**Definition 4.2.4** *Monitoring Region* $\zeta_s$ *for any subscriber* $s$ *located in cell* $C_{i,j}$ *may be calculated as,*

$$\zeta_s = C_{i,j} - \bigcup_{k=1}^{|m|} R(s, A_k), \tag{18}$$

*where* $R(s, A_k)$, $k \in [1...m]$, *denotes the spatial alarm regions relevant to subscriber* $s$ *intersecting the current subscriber cell* $C_{i,j}$.

## *4.3 Maximum Weighted Perimeter Rectangular Safe Region*

The safe region approach aims to reduce the number of alarm evaluations performed by the server. A rectangular shape has many properties required of the safe region as mentioned above. In this section, we discuss the *maximum weighted perimeter rectangular safe region* computation approach and present an algorithm for the same based on the concept of dynamic skylines [88].




(a) Steady Motion Assumption (b) Probability Density Function

**Figure 27:** Preliminaries for Safe Region Theorem

Figure 27(a) displays a mobile user at position $l_s(t)$; $l_s(t')$ is the previously recorded position of the client. The probability density function (pdf) for the client motion inside the safe region, denoted by $p(\phi)$, is given by:

$$p(\phi) = \begin{cases} \dfrac{1 + \frac{s}{t}\left\lceil \frac{\pi/2 - |\phi|}{s/t \cdot \pi} \right\rceil}{2\pi} & , \text{if } -\pi/2 \leq \phi \leq \pi/2 \\ \dfrac{1 - \frac{s}{t}\left\lceil \frac{|\phi| - \pi/2}{s/t \cdot \pi} \right\rceil}{2\pi} & , \text{otherwise} \end{cases}$$

In the above pdf formula, *s*, *t* are parameters of steadiness such that $s/t < 1$. Figure 27(b) displays the pdf for $s = 1$ and for different values of $t$. The value of $s/t$ determines the weight to be assigned to the probability of the client moving in the direction of its current motion. $t$ determines the granularity of change in $\phi$ for which the probability value decreases. As shown in Figure 27(b), the probability of the client moving in a direction such that $0 \leq \phi \leq \pi/t$ is the same; for values of $\phi > \pi/t$, this probability decreases. Assuming random direction of motion would lead to a probability of $1/2\pi$ for all values of $\phi$.

**Proposition 4.3.1** *Assume that an object moves in a direction $\phi$ as shown in Figure 27(a) with a speed $v$. Assuming a convex safe region $\Psi_s$, previous location $l_s(t')$ and an updated*

*location $l_s(t)$, we compute the average alarm evaluation cost $C_s$ over time as:*

$$C_s = C_l \cdot \Big( \int_{-\pi}^{\pi} \frac{r(\phi)p(\phi)\mathrm{d}\phi}{2\pi v} \Big)^{-1}$$

*In the above equation, $C_l$ is the cost of a single alarm evaluation, $\phi$ is the angle between the direction of motion of the object and its previously recorded direction of motion $\overline{l_s(t)l_s(t')}$, $r(\phi)$ is the length of the segment $\overline{l_s(t)R}$ where $R$ is the point on the safe region boundary at which the next alarm evaluation is expected to occur. Given the angle $\phi$, the elapsed time before the next evaluation is $r(\phi)/v$. The average elapsed time over all values of $\phi$ is given by $\int_{-\pi}^{\pi} \frac{r(\phi)p(\phi)\mathrm{d}\phi}{2\pi v}$. Therefore, we have,*

$$C_s = \frac{C_l \cdot 2\pi v}{\lambda(\phi)}, \tag{19}$$

*where $\lambda(\phi) = \int_{-\pi}^{\pi} r(\phi)p(\phi)\mathrm{d}\phi$ is the weighted perimeter of the safe region. In order to minimize the alarm evaluation costs, we have to maximize the value of the weighted perimeter. Therefore, the problem of minimizing alarm evaluation costs reduces to finding a rectangular safe region with maximum weighted perimeter.*

We now present an algorithm to compute the maximum weighted perimeter safe region for a user. Our algorithm applies the concept of *dominating point* and appropriate heuristics, to find the *four skyline points* [101] which form the corner points of the rectangular safe region. The algorithm accepts the current position vector $\vec{O}$ for a user and the current grid cell $\mathcal{G}(\vec{O})$ in which the user resides as inputs. The set of alarms intersecting the grid cell $\mathcal{G}(\vec{O})$ are considered for safe region computation. In case no relevant alarm regions intersect the grid cell $\mathcal{G}(\vec{O})$, the entire cell is returned as the safe region. Otherwise, the algorithm adopts a four step approach outlined below.

**Step 1: Determine Candidate Point Set.** The algorithm partitions the cell $\mathcal{G}(\vec{O})$ into four quadrants with current subscriber position $\{O_x, O_y\}$ as the origin. We define a set of *candidate points* $\mathcal{C}$ and a set of *tension points* $\mathcal{T}$ for each quadrant. The candidate point set is the set of points which can potentially form a corner point of the safe region. Tension

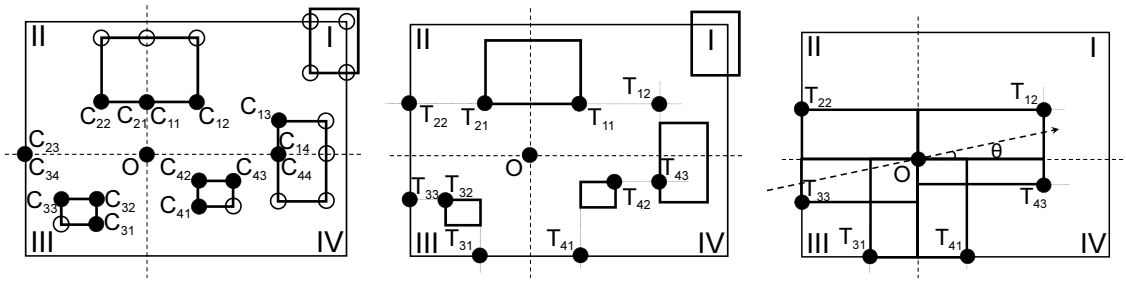
points are obtained from the set of candidate points by ensuring that only points that form largest possible rectangular regions not overlapping the spatial region of any alarm region are selected.

The set of candidate points is determined as follows. Firstly, the spatial region corner for each relevant alarm is selected as a candidate point in its appropriate quadrant. For alarm regions which do not completely lie inside the grid cell $\mathcal{G}(\vec{O})$, the intersection points of the grid cell boundary and the alarm region are selected as candidate points. Secondly, for alarm regions which intersect the x-axis or y-axis of the coordinate axes with origin at $\{O_x, O_y\}$, we also consider points of intersection of the alarm regions with the axes as candidate points. The algorithm trims the set of candidate points in the next step. Firstly, if multiple candidate points in a quadrant intersect the x-axis (or y-axis), all candidate points other than the point on the x-axis (or y-axis) closest to the origin are removed from the set $\mathcal{C}$. Further, we remove points which *fully dominate*[1] any other point in $\mathcal{C}$. Finally, the points are sorted according to increasing distance of their x-coordinate from the origin. Points with the same x-coordinate are arranged in order of decreasing distance of y-coordinate from origin.

**Step 2: Determine Tension Point Set.** The set of candidate points is then processed in the following manner to obtain the set of tension points. Each tension point $T_{Qi}$, where $Q \in \{1, 2, 3, 4\}$ represents the quadrant the point belongs to, is assigned the same x-coordinate as the corresponding candidate point $C_{Qi}$. $T_{Qi}$ is assigned the same y-coordinate as that of $C_{Qi-1}$, or $T_{Qi-1}$ if $T_{Qi}$ and $T_{Qi-1}$ have the same x-coordinate. The y-coordinate of $T_{Q1}$ is set as either the top bound of the cell or the y-coordinate of a candidate point intersecting the y-axis if any. Finally, duplicate points and points lying on the x-axis or y-axis are trimmed from the set $\mathcal{T}$.

**Step 3: Determine Component Rectangles.** The set of tension points form the opposite

[1] $P_1$ is said to fully dominate $P_2$, if $P_1.x > P_2.x$ and $P_1.y > P_2.y$.




(a) Candidate Point Set (b) Tension Point Set (c) Component Rectangles

**Figure 28:** Maximum Weighted Perimeter Safe Region Computation

corner (opposite to the origin) of the set of candidate *component rectangles* in each quadrant. The final safe region is composed of the intersection of the component rectangles from each quadrant. It seems logical to select component rectangles with the maximum perimeter in each quadrant as this will lead to safe regions with the maximum weighted perimeter.

**Step 4: Determine Safe Region from Component Rectangles.** Computation of the maximum weighted perimeter safe region can be involved and can lead to expensive computations. Our algorithm adopts greedy heuristics in order to quickly compose a suitable safe region from the set of component rectangles. As opposed to an optimal solution which enumerates every possible combination of component rectangles and computes the weighted perimeter for each combination thus taking quartic time, our approach performs greedy decisions. We first select the quadrant in which the *pdf* of the expected motion of the object is maximum. The component rectangle with the largest weighted perimeter in this quadrant is selected. Quadrants are further selected dependent on the distribution of *pdf* values in the quadrant using the steady motion assumption. At each step the component rectangle which forms the safe region with the largest weighted perimeter is selected. The algorithm continues until all four quadrants are processed using this greedy heuristic.

Figure 28 shows an example of our safe region computation approach. The candidate point set for the given scenario is as shown in Figure 28(a). The black dots represent the candidate points whereas the hollow dots represent points which are trimmed from the

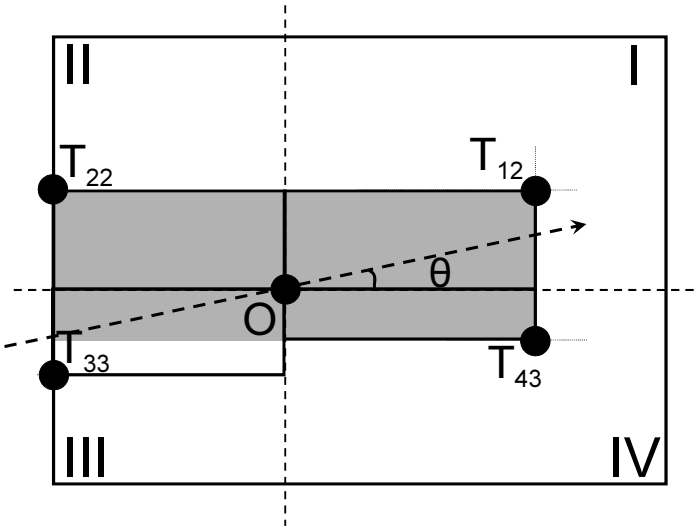



**Figure 29:** Final Safe Region

candidate point set as explained in Step 1 above. Figure 28(b) displays the set of tension points obtained from the candidate point set as explained in Step 2. Figure 28(c) displays the component rectangles formed by selecting a few of the tension points. The largest component rectangle in Quadrant I is first selected as the pdf values for object motion are maximum in this quadrant. Pdf values of object motion are expected to be next highest in Quadrant IV because the angle $\theta$ shown in Figure 28(c) is such that $\theta < \pi/4$.

If the value of $\theta$ was greater than $\pi/4$, Quadrant II would be selected first in stead of Quadrant IV. Addition of the component rectangle with tension point at $T_{43}$ provides a safe region with larger weighted perimeter compared to the safe region obtained by adding component rectangle with tension point at $T_{41}$. Finally, the component rectangles with tension points at $T_{22}$ in Quadrant II and $T_{33}$ in Quadrant III (in order of expected pdf values) are selected. The shaded region composed out of the component rectangles, as shown in Figure 29, forms the safe region.

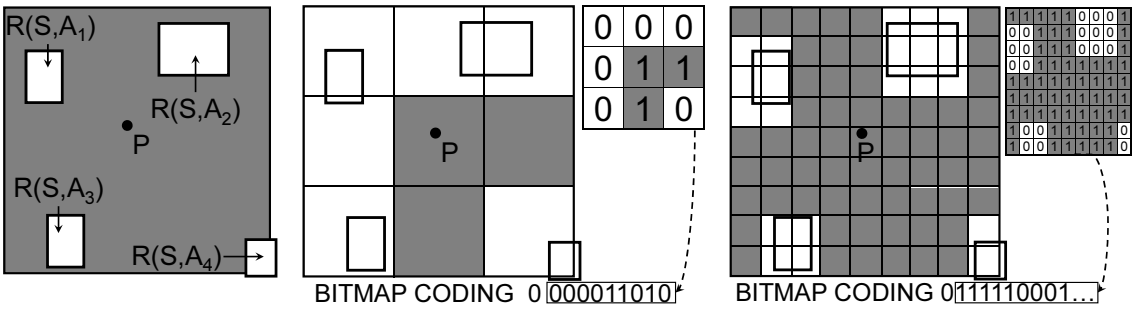
## *4.4 Bitmap Encoded Safe Region Computation*

Rectangular safe regions exhibit various advantages such as compact representation which leads to low server to client safe region communication cost and fast containment detection

at the client end. However, the rectangular shape restriction on safe region forces even powerful clients to underutilize their computational capacity. In this section, we relax the restriction on safe region shapes and consider rectilinear polygonal representations for safe region. We introduce *bitmap encoded safe region* (BSR) techniques for quickly and efficiently representing rectilinear polygons using bitmaps. This approach provides flexibility in safe region computation by providing larger, complex safe regions for powerful clients, thus personalizing the safe region for each client according to its computational capacity. Figure 30(a) displays the monitoring region (shaded region) for subscriber at point $P$ with four relevant alarm regions intersecting the grid cell. The server may compute the monitoring region as the safe region for the client and communicate it to the client. Note that the server is virtually pushing the relevant alarms onto the client in this scenario by providing this safe region. Each alarm region may be represented by the bottom-left and top-right corner point locations. We consider this as an *optimal approach* from the client perspective as the client has complete knowledge of all alarms in its vicinity. However, this approach may not be feasible from the point of view of communication costs incurred while broadcasting safe region to the clients. Additionally, for areas with high density the server may push a large number of alarms onto the client which would lead to heavy load for weak clients. To counter this problem, we now develop the concept of *bitmap encoded safe regions*, which provides an estimation of the actual safe region using a bitmap, allowing for trade-off between size of the bitmap and accuracy of safe region representation.

**Definition 4.4.1** *A bitmap encoded safe region represents a safe region $\Psi_s$ for subscriber $s$ using a bitmap $B$ of length $n$. A bit value of 1 indicates that a predefined region (cell) belongs to the safe region; whereas a 0 bit indicates the negation.*

We first describe a *Grid Bitmap Encoded Safe Region (GBSR)* computation technique and exhibit its inability to accurately and efficiently represent safe regions. An extension to this approach using the pyramid data structure [95], referred to as the *Pyramid Bitmap Encoded Safe Region (PBSR)* approach, allows us to represent safe regions accurately as well




(a) Monitoring Region (b) Grid Bitmap Encoding (3×3) (c) Grid Bitmap Encoding (9×9)

**Figure 30:** Grid Bitmap Encoded Safe Region Computation

as efficiently. BSR techniques exhibit the following advantages: (i) for low alarm density regions, it allows for further reduction of alarm evaluations compared to the rectangular safe region approaches, (ii) it supports varying granularity of safe region computations thus supporting heterogeneity among client capabilities, and (iii) clients can determine their position with respect to the safe region using a predefined (worst case) number of computations.

### 4.4.1 Grid Bitmap Encoded Safe Region Computation

The safe region for a subscriber $s$ can be represented by the set of grid cells as shown in Figure 30(b).

**Proposition 4.4.2** *We use a grid bitmap scheme to represent the safe region within the monitoring region shown in Figure 30(a). The cell $C_{i,j}$ is represented by a single bit $B(C_{i,j})$. If $C_{i,j} \bigcap \Sigma_{k=1}^{m} R(s, A_k) = \emptyset$ we set $B(C_{i,j}) = 1$ denoting that the entire cell $C_{i,j}$ belongs to the safe region $\Psi_s$, else we set $B(C_{i,j}) = 0$ and split $C_{i,j}$ into $U \times V$ smaller equi-sized cells. The same encoding procedure is used for each smaller cell. This bitmap encoding technique provides a compact representation for safe region $\Psi_s$.*

Figure 30(b) shows the safe region representation for the safe region of Figure 30(a) using a bitmap encoding scheme. No alarm regions intersect the three shaded cells which are represented by 1's; other cells intersecting with alarm regions are represented by 0's. The

safe region is represented using a simple bitmap $B = 0000011010$ which represents the cell bit values in a raster scan fashion. The first zero bit corresponds to the entire cell, indicating that the cell does not belong to the safe region and has intersecting alarm regions. As visible from Figure 30(b), this bitmap encoding is able to represent only a small portion of the monitoring region thus providing a poor estimate of the actual safe region. Figure 30(c) presents a 9×9 split of the cell at a finer resolution which allows for more accurate representation of the safe region. However, this approach is inefficient for the following two reasons: (i) it unnecessarily uses a much larger bitmap than required to represent the safe region, and (ii) different regions have different alarm densities thus making it difficult to select a uniform grid cell size. The PBSR approach overcomes these deficiencies by allowing for more accurate representations of the safe region while providing a smaller bitmap size.

### 4.4.2 Pyramid Bitmap Encoded Safe Region Computation

The pyramid representation splits cells in the *base* grid (level $L$=0) with $B(C_{i,j}^{L}) = 0$ *only* into $U \times V$ smaller cells, where $U$, $V$ are system defined parameters. The process may be further repeated for several iterations to form smaller cells at each level thus forming a pyramid data structure of height $h$. As shown using a pyramid structure with $h = 2$ in Figure 31, by further splitting cells with $B(C_{i,j}^{0}) = 0$ into a 3×3 grid we obtain a much more accurate representation for the safe region. Compared to the grid-based approach which either does not represent the safe region accurately (3×3 grid in Figure 30(b)) or computes a much larger bitmap (9×9 grid in Figure 30(c)), the PBSR approach provides flexibility in computation of the safe region. For example, the GBSR approach requires 82 bits, 1 bit for the entire cell and 81 bits for the 9×9 grid, to represent the safe region in Figure 30(c). In comparison the PBSR approach requires only 64 bits, 1 bit for the entire cell, 9 bits for the cells at level 1 and only 54 bits for the cells at level 2, to represent the same safe region as shown in Figure 31.

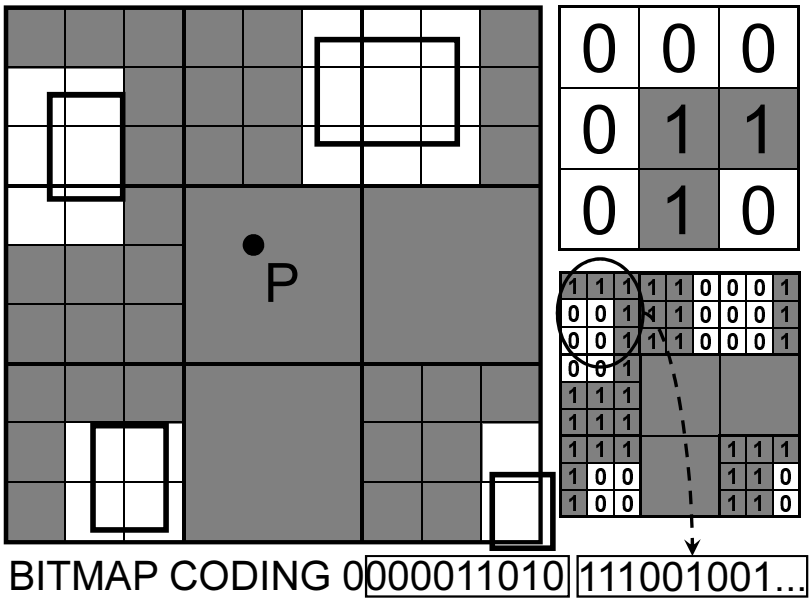



**Figure 31:** Pyramid Bitmap Encoded Safe Region

The bitmap $B$ is initially assigned a *null* value and current level $L$ of the pyramid is set to zero (line 2). A brief outline of the procedure is given below. The pyramid representation of the base cells is constructed for height $h$ by splitting cells iteratively into $U \times V$ cells. This step can be performed offline by the server thus providing a precomputed pyramid representation for safe region computation. Next, starting from the base cells (level $L = 0$) we determine if each cell intersects any relevant alarms. Cells not intersecting with any relevant alarm regions are assigned a bit value $B(C_{i,j}^{L}) = 1$ indicating that they are a part of the safe region; else a cell is assigned bit value 0. For cells at each level $L - 1$ ($L \leq h$) which have an assigned bit value 0, we consider the relevant $U \times V$ *children* cells at Level $L$ and assign a bit value 0 or 1 considering intersection of the cell with relevant alarms at each level of the pyramid.

**Proposition 4.4.3** *The PBSR approach for safe region computation allows us to represent the safe region $\Psi_s$ in terms of a bitmap of size $|B|$. The height of the pyramid $h$ allows us to control the accuracy of representation of the safe region at the cost of computing a larger bitmap for more accurate representations.*

**Algorithm 1**: Pyramid Bitmap Encoded Safe Region Computation

**Input**: $C^0_{k,l}$, $h$, $U$, $V$, $A^{rel}_s$
**Output**: $B$

$C^0 \leftarrow \{C^0_{k,l}\}$;
$B = null$; $L = 0$;
**while** $(L < h)$ **do**
  $C^{L+1} \leftarrow SPLIT(C^L, U, V)$;
  $L = L + 1$;
**end**
$L = 0$;
**while** $(L \leq h)$ **do**
  **for** $(i = (k-1) \cdot U^L + 1;\ i \leq k \cdot U^L;\ i++)$ **do**
    **for** $(j = (l-1) \cdot V^L + 1;\ j \leq l \cdot V^L;\ j++)$ **do**
      **if** $((L = 0)\ \|\ ((L \neq 0)\ \&\&\ (B(C^{L-1}_{\lfloor i/U \rfloor, \lfloor j/V \rfloor}) = 0)))$ **then**
        **if** $(C^L_{i,j} \bigcap A^{rel}_s = \emptyset)$ **then**
          $B(C^L_{i,j}) = 1$;
        **else**
          $B(C^L_{i,j}) = 0$;
        **end**
        $B = B\ ||\ B(C^L_{i,j})$;
      **end**
    **end**
  **end**
  $L = L + 1$;
**end**

We now define *Coverage* and *Bitmap Size* which allow us to control the quality of the safe region representation for our BSR computation techniques.

**Definition 4.4.4** *The coverage of a safe region representation $\Psi_s$, denoted by $\eta(\Psi_s)$, is defined as the ratio of area of the safe region using the BSR representation to the area of the monitoring region.*

$$\eta(\Psi_s) = \frac{\sum_{L=0}^{h} \sum_{i=(k-1)\cdot U^L+1}^{k\cdot U^L} \sum_{j=(l-1)\cdot V^L+1}^{l\cdot V^L} \frac{\alpha \cdot \beta}{(U \cdot V)^L} \cdot B(C^L_{i,j})}{\zeta_s}, \tag{20}$$

where $\alpha$, $\beta$ defines the size of a base grid cell $C^0_{i,j}$ of the pyramid.

**Definition 4.4.5** *The bitmap size for safe region $\Psi_s$, denoted by $\vartheta(\Psi_s)$, is defined as the*

*number of bits in the BSR representation of the safe region.*

$$\vartheta(\Psi_s) = 1+ \sum_{L=0}^{h-1} \sum_{i=(k-1)\cdot U^L+1}^{k\cdot U^L} \sum_{j=(l-1)\cdot V^L+1}^{l\cdot V^L} (1 - B(C_{i,j}^L)) \cdot U \cdot V \quad (21)$$

In practice, we want to achieve high coverage with as small bitmap size as possible. Each client may specify the maximum height of the pyramid used by the PBSR approach for computing its safe region. In the worst case scenario, the client may need to determine its position relative to the safe region at each level of the pyramid data structure.

For the BSR approach, safe region for a client needs to be recomputed only when the client moves out of the base grid cell. Note that a client may move out of its safe region without triggering any relevant alarms even while it is inside the grid cell. No safe region recomputation needs to be performed in such situations for the BSR approach. In case the client triggers an alarm on moving outside its safe region but stays within the cell $C_{i,j}^0$ corresponding to the safe region, the safe region can be quickly updated by considering the triggered alarm to be a part of the safe region. Additionally, BSR approaches can be optimized by precomputing the bitmap at each pyramid level for public alarms.

#### *4.4.2.1 Client Safe Region Containment Detection*

The MPSR approach demands that the client monitor its position within rectangular shape safe region which requires simple computations on part of the client. For the PBSR region approach, the client needs to determine its position with respect to the safe region from the bitmap $|B|$. The client determines its position at each level of the pyramid in order to determine if it is within the safe region or not. In the worst case scenario, the client needs to perform $h$ computations, one at each level of a pyramid of height $h$; on an average it will perform much fewer than $h$ computations. Algorithm 2 outlines the client safe region containment logic required for the PBSR approach.

The algorithm accepts as input the bitmap $B$ and the position vector $\vec{p_s}$ for subscriber $s$. The algorithm returns the containment detection result $CDR$ indicating a value true if client

**Algorithm 2**: PBSR Client Safe Region Containment Detection

```
   Input: B, p_s
   Output: CDR ∈ {true, false}
 1 L = 0; LStartIndex = 0; LEndIndex = 0; posIndex = 0;
 2 numFalsePrevL = 0; numFalsePrevLPosIndex = 0;
 3 while (LStartIndex ≠ |B|) do
 4     if (B[posIndex] == true) then
 5         return true;
 6     else
 7         L = L + 1;
 8         for (i = LStartIndex; i ≤ LEndIndex; i++) do
 9             if (B[i] == false) then
10                 if (i < posIndex) then
11                     numFalsePrevLPosIndex++;
12                 end
13                 numFalsePrevL++;
14             end
15         end
16         cellid = getRelCellPos(p_s, L);
17         LStartIndex = LEndIndex + 1;
18         LEndIndex+ = (numFalsePrevL · U · V);
19         posIndex = LStartIndex + cellid + (numFalsePrevLPosIndex · U · V) − 1;
20         numFalsePrevL = 0; numFalsePrevLPosIndex = 0;
21     end
22 end
23 return false;
```

lies inside safe region or false if client lies outside safe region. Initially the level $L$ of the pyramid is set to zero. The algorithm also identifies the start index $LStartIndex$ and end index $LEndIndex$ for bitmap values in $B$ related to level $L$. We define the set of bitmap values from start to end index for any level $L$ as a $block$. The bitmap index concurrent to the cell the client currently belongs to is indicated by $posIndex$ (line 1). Additionally, the algorithm needs to keep a count of the number of $false$ bits $numFalsePrevL$ in a block and the number of false bits $numFalsePrevLPosIndex$ before the $posIndex$ in a block (line 2). The algorithm checks for each level $L$, if the bitmap value is $true$ indicating that client location lies within the safe region (lines 4-6). Otherwise the algorithm increments the level $L$ and maintains the count of number of false values in previous block and the number of false values before $posIndex$ in previous block of the bitmap (lines 8-15). These values are used to determine the $LStartIndex$, $LEndIndex$ and $posIndex$ in the new block corresponding to the next level of the pyramid in the bitmap (lines 16-19). This

computation is repeated for each level $L$ of the pyramid to determine if the subscriber lies within the safe region (lines 6-21).

In order to facilitate installation of new alarms, the server maintains a main memory grid index on the safe region of all clients. Location updates are required of all clients whose safe region intersects the new spatial alarm region. Spatial alarm information is indexed using a disk resident R-tree structure. We use a 3-dimensional R-tree which indexes the subscriber relevance information for private, shared and public alarms as well as the bottom-left and top-right points of the safe region *minimum bounding rectangles (MBRs)*.

## *4.5 Experimental Evaluation*

In this section, we evaluate the performance of our safe region computation techniques using three different sets of experiments. The first set of experiments performs an evaluation of the maximum weighted perimeter rectangular safe region approach. The second experiment evaluates the bitmap encoded safe region (BSR) approaches, namely grid bitmap encoded safe region (GBSR) and pyramid bitmap encoded safe region (PBSR). The final experiment provides an evaluation of the safe region techniques compared to periodic processing (PRD), safe period-based (SP) computation [28] and the optimal (OPT) approach as described in beginning of Section 4.4. The optimal approach does not consider any restrictions on resource availability and assumes all relevant alarms within the monitoring region are pushed to the client, which implies the client is fully aware of all relevant alarms in its vicinity. We measure the performance of all approaches on different evaluation metrics like number of client-to-server messages, downstream server-client bandwidth consumption, client energy consumption and server processing time. We do not measure alarm trigger accuracy as the parameters adopted for each processing approach *ensure 100% of the alarms are triggered in all scenarios*. The sequence of alarms to be triggered is determined by a very high frequency trace of the motion pattern of the vehicles.

### 4.5.1 Experimental Setup

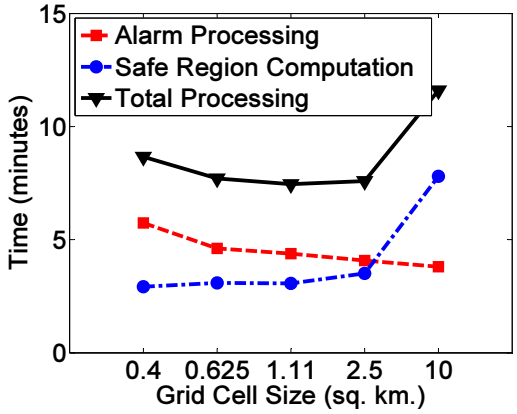
Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS [8]). Vehicles are randomly placed on the road network according to real world traffic densities as given in [27]. The simulator simulates the motion of vehicles on roads with appropriate velocity information; at intersections, vehicles may move in any direction with attached probability values. We use a map of Atlanta and surrounding regions, which covers an area around 1000 $km^2$ in expanse, to generate the trace. Our experiments use traces generated by simulating vehicle movement for a period of one hour, results are averaged over a number of such traces. Default traffic volume values allow us to simulate the movement of a set of 10,000 vehicles on the above road network. Each vehicle generates a set of position parameters during the simulation; position information is evaluated against installed spatial alarms indexed in a $R^*$-tree [31]. The default spatial alarm information consists of a set of 10,000 spatial alarms installed on alarm targets distributed uniformly over the entire map region. We vary the fraction of private, shared and public alarms installed in order to vary the number of alarms relevant to each subscriber. Default setup assumes 10% of the alarms are public alarms; private and shared alarms are present in the system in the ratio 2:1. This simulator setup allows us to the test the robustness of our framework under realistic mobility patterns.

### 4.5.2 Experimental Results

**Performance of Maximum Weighted Perimeter Rectangular Approach.** This set of experiments is designed to study the performance of the maximum weighted perimeter rectangular safe region approach as we vary the parameters of steadiness $s, t$. The results for $s = 1$ and different values of $t$ in comparison with a non-weighted approach (no steady motion assumption) are shown in Figure 32. The non-weighted perimeter approach improves upon the approach presented in [57] by allowing for overlapping alarm regions.

| Cell Size (sq. km.) | Non – Weighted | s=1, t=4 | s=1, t=16 | s=1, t=32 |
|---|---|---|---|---|
| 0.4 | 1.752 | 1.737 | 1.735 | 1.733 |
| 0.625 | 1.629 | 1.608 | 1.607 | 1.606 |
| 1.11 | 1.509 | 1.488 | 1.485 | 1.484 |
| 2.5 | 1.418 | 1.395 | 1.391 | 1.39 |
| 10 | 1.382 | 1.348 | 1.343 | 1.34 |

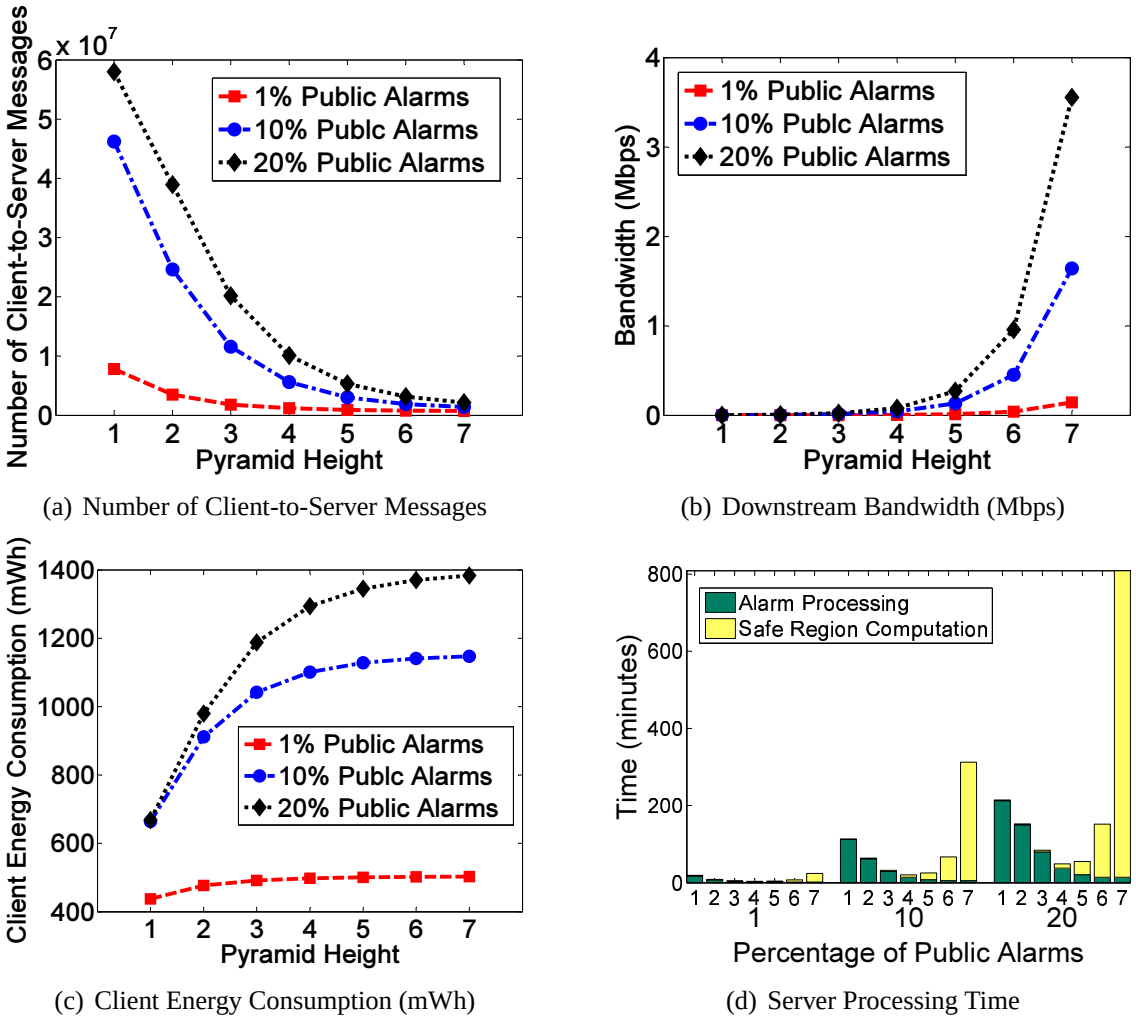
(a) Number of Client-to-Server Messages (in millions)




(b) Server Processing Time for Weighted Perimeter Approach (s=1, t=32)

**Figure 32:** Performance of Rectangular Approach

The approach presented in [57] leads to alarm misses and erroneous safe regions in such scenarios. Figure 32(a) shows the number of client-to-server location messages exchanged with different grid cell sizes. The weighted perimeter approach consistently performs better than the non-weighted perimeter approach even though by a small margin. Considering the fact that more than 60 million location messages are produced for each trace, it can be observed that less than 3% of messages need to be communicated to the server using any of the rectangular safe region approaches. The other observation from this figure is that with increasing grid cell size the number of client-to-server messages reduces. This is as expected because with larger grid cell sizes larger safe regions are computed and the client stays within the safe region for a longer duration. Figure 32(b) shows the server processing time as we vary the size of the grid cell. As grid cell size is increased, alarm processing costs decrease due to the smaller number of location messages being processed against the spatial alarm index. The safe region computation costs increase with increasing grid cell size due to larger number of intersecting alarms being considered for safe region computation. The total server processing time is minimum with a grid cell size of 2.5 sq. km. This value of grid cell size is used for further experimentation.




(a) Number of Client-to-Server Messages

(b) Downstream Bandwidth (Mbps)

(c) Client Energy Consumption (mWh)

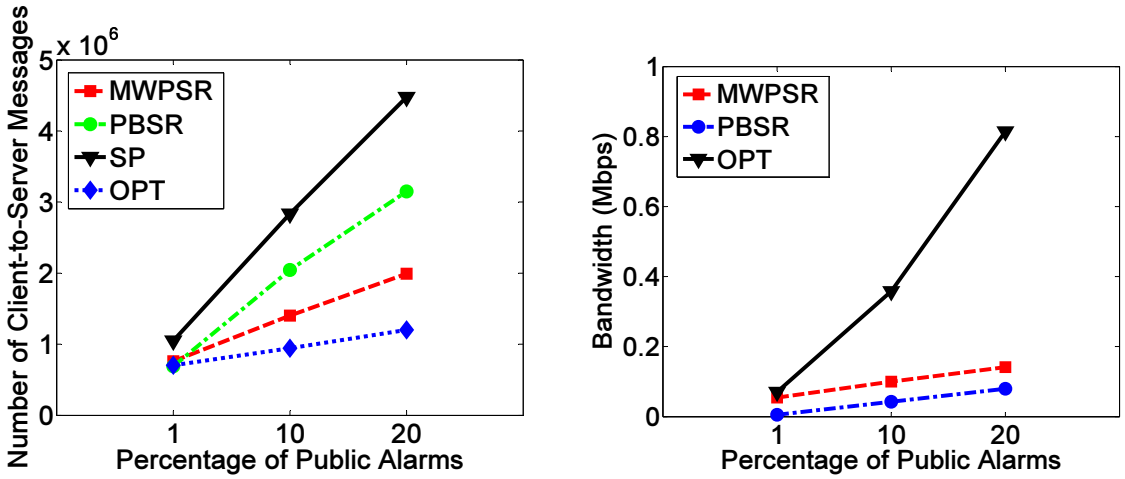
(d) Server Processing Time

**Figure 33:** Performance of BSR Approach

**Performance of BSR Approach.** This set of experiments is designed to evaluate the performance of the BSR approach. We vary the height of the pyramid from $h = 1$ (for GBSR) to $h = 7$ and observe the performance as shown in Figure 33. Figure 33(a) displays the number of client-to-server messages communicated as we increase the pyramid height from $h = 1$ to $h = 7$. It can be observed that the GBSR approach is highly inefficient as it limits safe region computation to a very high granularity. The safe region computed using this approach provides a very coarse representation of the actual safe region forcing the clients to frequently send location messages as a result of which GBSR approach incurs high

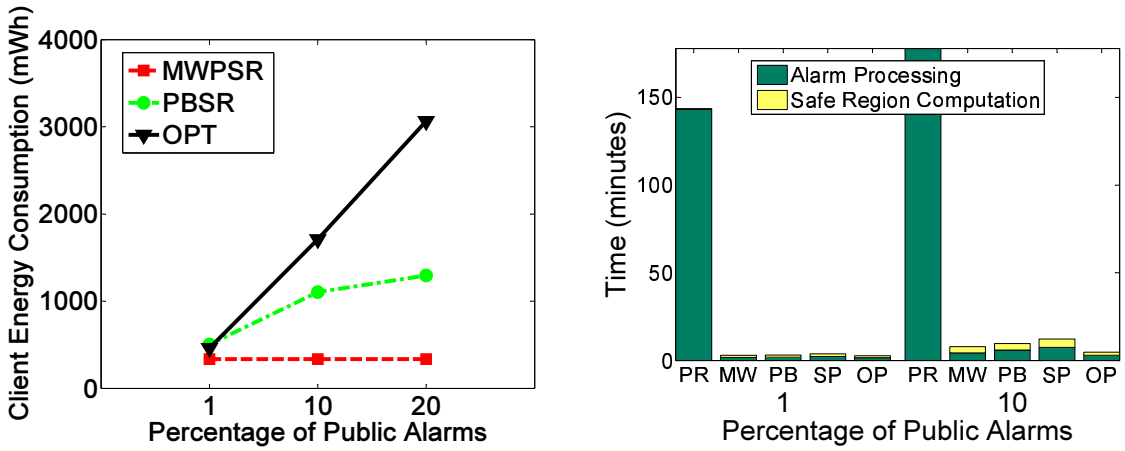
communication costs. As we increase the pyramid height, more accurate safe region representations can be computed and consequently number of messages transmitted experiences a sharp drop. Another observation is that BSR approaches display high sensitivity to alarm density levels; the performance deteriorates sharply for higher percentage of installed public alarms. On the other hand, the downstream bandwidth required by the server to broadcast the safe regions to the clients increases with pyramid height (Figure 33(b)). For higher pyramid levels, larger bitmaps are required to represent the safe region and hence higher bandwidth is required. For pyramid height $h = 7$, with high alarm density the downstream bandwidth requirement goes up to 3.5 Mbps, but for $h = 5$ this value remains below 200 Kbps even when percentage of public alarms is increased to 20%. Figure 33(c) displays the client energy consumption (in milliwatt-hours) used to determine client position within the safe region. For the GBSR approach the clients need to perform an average of 2-3 safe region containment detections per second resulting in low energy consumption. This cost does not experience a significant increase with pyramid height for low percentage of public alarms. For higher public alarm percentages, 6-7 safe region containment detections per second are required for a pyramid of height $h = 7$ resulting in higher energy consumption. As seen from Figure 33(d), for low pyramid height, safe region computation costs are low as relatively simpler computations are involved. On the other hand, alarm processing costs are high as a large number of messages are received from clients. On increasing pyramid height, alarm processing costs drop due to fewer client-to-server messages. The safe region computation costs increase due to high complexity of safe region computation. Even despite the fewer number of safe region computations being performed at higher pyramid height, the increase in cost of a single safe region computation is such that a net increase in safe region computation load is experienced. However, this cost can be significantly offset by using precomputed bitmaps for public alarms as mentioned earlier. For $h = 4$ or $h = 5$, the overall processing load is at its lowest point.

**Performance Comparison of Safe Region with Other Approaches.** Now we compare




(a) Number of Client-to-Server Messages

(b) Bandwidth (Mbps)




(c) Client Energy Consumption (mWh)

(d) Server Processing Time

**Figure 34:** Safe Region vs. Other Approaches

the performance of the maximum weighted perimeter rectangular safe region approach (MWPSR) and the pyramid bitmap safe region (PBSR) approach ($h = 5$) with periodic evaluation (PRD), safe period-based processing (SP) and the optimal approach (OPT) as described at the beginning of Section 4.4. As can be seen from Figure 34(a), the safe region approaches transmit few client-to-server messages. Periodic processing requires clients to transmit each location update to the server amounting to 60 million messages and is not shown in the figure. The safe period approach experiences significantly higher communication costs, approximately 2-3 times the cost incurred by the safe region approaches. This is due to the pessimistic assumptions required to ensure that the safe period approach

triggers all alarms with a 100% success rate. The optimal approach would require clients to transmit updates only when the spatial constraints for one or more relevant alarms are met and transmits fewest number of messages. Figure 34(b) displays the downstream bandwidth consumed by the system to broadcast safe regions to the clients. Safe period approach would also require that a computed safe period be broadcast to each client; however, we exclude the bandwidth incurred for this approach from these results. As expected the safe region approaches incur much lower bandwidth expense when compared to an optimal solution. PBSR ($h = 5$) performs the best for different percentage of public alarms in the system. Not surprisingly, client energy consumption for the optimal approach is significantly higher than the safe region approaches (Figure 34(c)) as the optimal solution is based on the assumption that clients have very high capacity. PBSR and MWPSR approaches lead to lower client energy consumption especially at higher alarm density levels. The processing load experienced by each approach is as shown in Figure 34(d). Periodic approach (PR) has much higher alarm processing costs as each update needs to be processed by the client and the server load does not scale. The processing load does not rise much at higher alarm densities as each update is processed by this approach for all percentage of public alarms. The MWPSR and PBSR approaches (MW and PB in the figure) experience lower server load due to much lower alarm processing load. With increasing percentage of public alarm values, the safe region computation as well as the alarm processing load rises; however, the total load incurred by the system is much lower than the periodic approach for all configurations. The PBSR approach again shows similar trends as the MWPSR approach; however, the CPU load incurred by this approach at higher alarm density levels is higher than MWPSR approach. The safe period (SP) approach experiences higher CPU load compared to the safe region approaches. This is a direct result of the larger number of updates that need to be processed by the safe period approach. Results for the optimal approach (OP) are plotted to show that the safe region approaches do not incur much higher CPU load except for the highest percentage of public alarms.

## *4.6 Related Work*

In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand [30, 76]. In addition to monitoring continuously changing user information needs, spatial alarm processing systems also need to deal with the complexity of monitoring user location data in order to trigger relevant alerts in a non-intrusive manner.

Periodic reevaluation is commonly used for continuous monitoring of moving objects [60, 83, 93, 115]. Some work exists on monitoring continuous queries which applies the concept of safe region directly [35, 57, 93] or indirectly [114, 116]. Spatial alarms differ from this work as they do not demand periodic evaluation or reevaluation like continuous queries; in stead they require one shot evaluation which should result in a trigger when the alarm conditions are satisfied. Our work is focussed on determining the opportune moment for evaluating spatial alarms relevant to a client by seeking cooperation at the client end.

None of the previous work except [57] presents clear algorithms for safe region computation. The maximum weighted perimeter rectangular safe region approach outperforms the approach presented in [57]. Further, unlike our approach presented in Section 4.3, the algorithm presented in [57] cannot handle overlapping alarm regions, alarm regions overlapping multiple grid cells or alarm regions intersecting the axes of the coordinate system. Most importantly, previous work fails to consider an environment supporting heterogeneous clients. Our BSR techniques provide an elegant solution for exploiting client heterogeneity further easing the computational load on the server.

# CHAPTER V

# SCALABLE INFORMATION MONITORING FOR MOBILE SYSTEMS WITH NON-SPATIAL ATTRIBUTES

## *5.1 Introduction*

Availability of cheap location sensing devices and advancement in wireless communication technology has led to an explosion of location-based services. Other technological advances like large-scale deployment of sensor networks and information monitoring systems (e.g. traffic monitoring and analysis) have led to the availability of large amounts of context-aware location-sensitive information. Users can explore their environments for useful information from the convenience of GPS equipped vehicles or their cell phones. In this chapter, we introduce *location-centric triggers*, which provide useful means of allowing users to express their location-based information needs. Many information needs may be expressed using *location-centric triggers*, which require a one-shot evaluation, as opposed to the continuous re-evaluation required by the continuous query mechanism.

In monitoring systems, information is typically delivered to centralized servers in the form of data streams that arrive continuously, rapidly and in real time [52, 84]. It is not possible to control the order of arrival of data updates or to store all the data updates. A mobile information monitoring system is characterized by a large number of such data streams, some delivering positioning information for a large number of mobile users and others delivering large amounts of location-sensitive monitored data. Examples of such location-sensitive monitored data may include gas prices at gas stations, pollution levels at locations of interest or traffic conditions at major junctions.

We introduce the SLIM[1] system for efficiently monitoring information comprising of

[1] SLIM is an acronym for ScaLable Information Monitoring

spatial as well as non-spatial data. To the best of our knowledge, the SLIM system is the first to study the effects of considering the interaction between spatial and non-spatial data in mobile systems. In our system, users express their information needs in terms of location-centric triggers with spatial and non-spatial predicates. For example, a user may install a trigger of the form *"inform me when I am within one mile of gas station G and the price of gas is below $4"*. The central server receives and processes mobile position updates and monitored data updates from a large number of sources to determine the opportune moment to activate triggers. As opposed to current systems which focus on efficient evaluation of different types of continuous queries (e.g. kNN query, range query) based on spatial predicates alone ( [57, 58, 83, 113]), we consider a mobile system with a large number of location-sensitive attributes being delivered from multiple data monitoring sources. Introduction of non-spatial predicates renders existing solutions incapable of handling the mobile information monitoring problem efficiently. Addition of non-spatial attributes to the mix presents opportunities to perform optimizations beyond those possible with spatial attributes alone as explained in our motivating example (cf. Section 5.2).

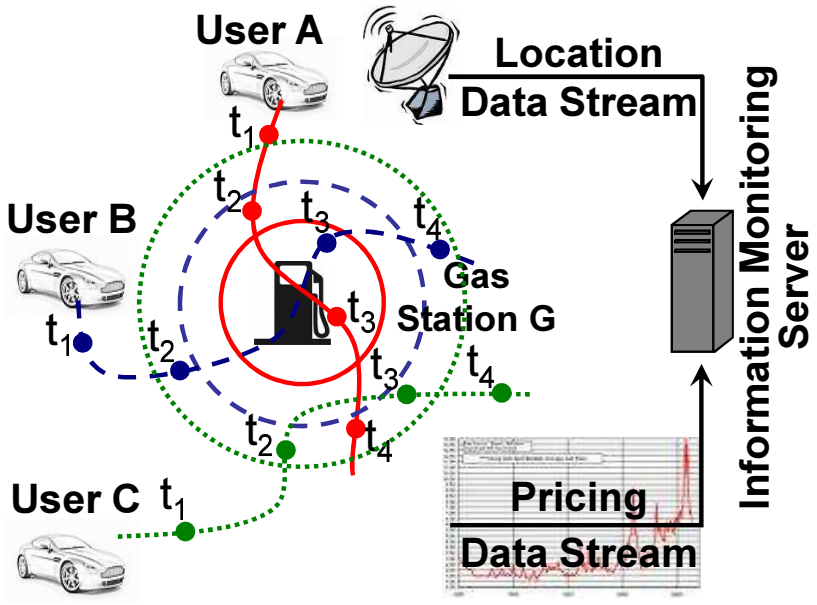
Location-centric trigger processing requires meeting three demanding objectives: (i) high accuracy, which ensures no triggers are missed, (ii) server scalability, which guarantees that the information monitoring server scales to large number of triggers, growing base of mobile users and a large number of data streams, and (iii) immediate evaluation of data updates in order to activate triggers at the earliest possible moment. Activation of location-centric triggers typically transforms into some action on part of the user, which is dependent on the presence of the user at the *location of interest* associated with the trigger, thus, making it imperative to activate triggers as soon as the spatial and non-spatial predicate conditions are met.

Solutions in the domain of streaming systems provide optimizations which lead to approximate answers [39, 106] or delayed evaluation as in the case of batch processing of data updates [22, 75]. Both are unable to meet our objectives of immediate evaluation with

100% accuracy. A simple solution which meets these objectives processes each and every data update on arrival to determine if any triggers are activated. However, this *in-transit processing* approach hurts the scalability of the system. Load shedding approaches [104] allow controlled tuning of the server load by dropping data updates randomly or selectively but cannot guarantee 100% accuracy.

The SLIM approach takes into account interaction between various attributes which facilitates dropping updates that previous approaches cannot avoid. SLIM selectively evaluates received data updates against installed trigger information, dropping updates which have *zero probability of activating any installed triggers* (cf. Section 5.3). In order to achieve 100% accuracy while selectively evaluating data updates, we introduce the concept of *safe containment*. Safe value containers are computed for each object in SLIM allowing the system to drop updates which lie within their respective safe value containers without evaluation. Additionally, safe value containers are communicated back to relevant participating objects in the system seeking their cooperation in the monitoring process and thus allowing for communication cost savings. However, safe value container computation leads to additional processing cost at the server.

We present efficient algorithms for safe value container computation for single-dimensional and multi-dimensional data. For single-dimensional data, we compute single safe value containers or multiple safe value containers. A comparative study of single versus multiple safe value containers reveals that multiple safe value containers are preferable when rate of change of data values is high and update frequency of data sources is low. We extend the concept of safe value containers to multi-dimensional data and discuss an algorithm for computation of two-dimensional safe value containers. Extension to multi-dimensional data ($> 2$) is also discussed briefly. Our experimental evaluation shows that server scalability is enhanced multiple times by deployment of safe containment techniques. Mobile clients may gain in terms of energy and bandwidth consumption by monitoring their position within their respective two-dimensional safe value containers.




| User | Distance | Price |
|---|---|---|
| A | 1 mile | < $4 |
| B | 2 miles | < $4.2 |
| C | 3 miles | < $4.5 |

| | |
|---|---|
| $t_2$ | $4.6 |
| $t_4$ | $4.3 |

**Figure 35:** Motivating Example

The rest of the chapter is structured as follows. Section 5.2 provides a motivating example. We define the fundamental concepts and notations associated with our system in section 5.3. Section 5.4 provides algorithms for computing single-dimensional and multi-dimensional safe value containers. This is followed by an experimental evaluation using a real world road network in section 5.5. Section 5.6 provides a brief discussion of related work.

## *5.2 Problem Motivation*

We motivate the optimization opportunities presented by addition of less dynamic non-spatial attributes to the information monitoring problem in mobile systems with the aid of an example. Figure 35 displays mobile users *A*, *B* and *C* with installed location-centric triggers on gas station *G*. Each user specifies spatial as well as non-spatial predicate conditions for her location-centric trigger. For example, the location-centric trigger for user *A* requires

that the trigger should be activated when the *user is within one mile of the gas station G* and *gas price is below $4*. Similarly, users *B* and *C* specify different spatial and non-spatial predicate conditions for their location-centric triggers. For the sake of exposition, assume that the rate at which the pricing data stream is delivered at the server is half the rate at which mobile clients update their positions. Further, we consider that the updates from the data streams are synchronized for the sake of simplicity. In the example, the location data stream comprises of mobile positioning information between time instants $t_1$ and $t_4$ (at one minute intervals) as shown in Figure 35. Similarly, pricing data are received at the server at time instants $t_2$ and $t_4$. A simple approach would process the data updates as and when they are received. However, as we discuss next, it is unnecessary for the server to process all received data updates.

Firstly, we observe that user *A* lies outside the spatial region associated with her installed trigger at time instants $t_1$, $t_2$ and $t_4$. If it was possible for the server to determine that the probability of user *A* being positioned within the spatial region of her installed trigger is zero at these time instants, the location data for the user at these time instants can be dropped without processing. Alternately, the user can save on precious energy and bandwidth costs by avoiding sending updates to the server at these time instants. Secondly, we observe that the pricing data at time instant $t_2$ does not satisfy the non-spatial constraints for any of the installed triggers. If the server can determine that the probability of the pricing data constraint being satisfied for any of the installed triggers on the gas station is zero at $t_2$, this update may be dropped. Thirdly, the position update for all mobile users at time instant $t_3$ can be dropped if the server recognizes the fact that even though the spatial constraints are satisfied for all the triggers, the probability of the non-spatial constraints being satisfied is zero due to the previous pricing data update at $t_2$. Lastly, the pricing data update at time instant $t_4$ may be dropped, even though it satisfies the non-spatial constraints for the trigger installed by user *C*, if the server recognizes that the probability of any of the users satisfying the spatial constraints for their respective triggers is zero at this time instant. The

last two cases highlight the benefits of considering the interaction between the different attributes involved in trigger processing. Keeping these observations in mind, we develop the concept of safe containment for *SLIM*. But first, we describe the fundamental concepts associated with our system.

## *5.3 Fundamental Concepts and Notations*

In this section, we describe the data model, location-centric trigger model and the processing model adopted by our system. Each *mobile user* $\mathfrak{u}_i \in \mathcal{U}$ expresses her location-based information needs in the form of *location-centric triggers* $\mathfrak{t}_{i,j} \in \mathcal{T}$ at a *location of interest* $\mathfrak{l}_j \in \mathcal{L}$. The triggers are installed at the server which receives updates from multiple data sources; the location-centric characteristic of our system implies that sources delivering location updates from mobile users are a consistent feature of the system. The server also receives updates from data sources associated with other attributes and processes them to determine if any relevant triggers need to be activated.

**Data Model**: Each data source delivers tuples of the form $S_k(o(t)) = \langle o, t, a_1, a_2, ..., a_n \rangle$, containing $n - dimensional$ data. $a_r$, $r \in [1...n]$, is a value in domain $D_r$ representing the value of the $r^{th}$ attribute, associated with the object $o \in \mathcal{O}$ and delivered by source $S_k$, at time $t$. The data sources deliver content associated with either a mobile user or a location of interest in the system, hence, $\mathcal{O} = \mathcal{U} \bigcup \mathcal{L}$. For example, a data source delivering positioning information for a large number of mobile clients delivers tuples of the form $S(\mathfrak{u}_i(t)) = \langle \mathfrak{u}_i, t, x, y \rangle$, which indicates the position of user $\mathfrak{u}_i$ in the two-dimensional coordinate space at time $t$. A monitored data source may deliver tuples of the form $S(\mathfrak{l}_j(t)) = \langle \mathfrak{l}_j, t, price \rangle$, which represents the attribute $price$ (of gas) at location of interest $\mathfrak{l}_j$ (gas station) at time $t$.

**Trigger Model**: A location-centric trigger specifies a set of spatial predicates and non-spatial predicates in the form $\langle attribute \rangle$ $\langle op \rangle$ $\langle value \rangle$ where $\langle op \rangle \in \{<, >, \leq, \geq\}$, combined using the $\wedge$ logical operator. For example, the trigger installed by user $A$ on gas station $G$ in the previous section can be expressed as $\mathfrak{t}_{A,G} = (x \geq -\sqrt{2} \wedge x \leq \sqrt{2} \wedge$

$y \geq -\sqrt{2} \wedge y \leq \sqrt{2} \wedge p < 4)$. The first four constraints express the spatial trigger region using the *minimum bounding rectangle (MBR)* of the circle of radius one mile around the gas station $G$, assuming that it is located at the origin of the coordinate space. The last predicate condition expresses the requirement on gas price as specified by the user. The predicate conditions specified on the spatial region will be a common feature of all triggers; however, different locations of interest will have different monitored attributes associated with them. We assume that each trigger specifies predicate conditions on all the monitored attributes associated with its corresponding location of interest.

We classify location-centric triggers into three different categories depending on the relevance to the subscriber population: *private*, *shared* and *public* triggers. Consider a trigger processing system with $\mid \mathcal{U} \mid$ users. Private triggers $t_{i,j} \in \mathcal{T}$ are relevant to a single user, where $i \in [1... \mid \mathcal{U} \mid]$. Shared triggers $t_{\mathcal{U}',j} \in \mathcal{T}$, where $\mathcal{U}' = \{i_1, i_2, ..., i_{u'}\}$ and $u' \geq 2$, are relevant to at least two users. $u'$ specifies system limitations on maximum number of users permitted to share a trigger. Public triggers $t_{\mathcal{U},j} \in \mathcal{T}$ are deemed to be relevant to the entire subscriber base. An additional constraint specifies that a user $u_i$ may have only one trigger relevant to a given location of interest $l_j$.

**Processing Model**: Our processing model is intended to outperform the in-transit processing and load shedding approaches [104, 49] in order to provide a scalable mobile system with 100% trigger activation success rate. We now model our *selective processing* approach which drops all data updates with zero probability of activating relevant triggers without trigger processing.

Consider a trigger $t_{i,j}$ which has predicate conditions associated with data being delivered by $m$ different data sources. The probability of a data update from the $k^{th}$ data source $S_k(o(t))$, being able to activate this trigger, denoted by $Pr[S_k(o(t)) \Rightarrow t_{i,j}]$, is dependent upon two factors: (i) the probability of the data update being able to satisfy its predicate constraints associated with the trigger. We represent the constraints by the range of values $R^k_{i,j}$ and denote the associated probability by $Pr[S_k(o(t)) \in R^k_{i,j}]$. (ii) the probability of all

previous data updates $S_{k'}(o'(t'))$, where $k' \in [1...m]$ and $k' \neq k$, satisfying their respective predicate conditions specified by the trigger $\mathfrak{t}_{i,j}$. The probability of any data update satisfying its predicate conditions for trigger $\mathfrak{t}_{i,j}$ is denoted by $Pr[S_{k'}(o'(t')) \in R_{i,j}^{k'}]$. Here $t'$ denotes the time instant at which the latest data update from $S_{k'}$ for object $o'$ was received by the server, such that $t' < t$.

Now, we consider the processing model for a large number of triggers associated with an object $o$. Consider the set of triggers $\mathcal{T}' \subset \mathcal{T}$ relevant to the object $o$. Any data update associated with this object should be processed at the server, if the probability of activating at least one trigger, denoted by $Pr[S_k(o(t)) \Rightarrow_{\geq 1} \mathcal{T}']$, is greater than zero. Thus, we have,

$$Pr[S_k(o(t)) \Rightarrow_{\geq 1} \mathcal{T}'] = 1 - Pr[S_k(o(t)) \not\Rightarrow \mathcal{T}'], \tag{22}$$

where $Pr[S_k(o(t)) \not\Rightarrow \mathcal{T}']$ denotes the probability of not activating any relevant triggers in $\mathcal{T}'$. Now, we separately consider the case where object $o$ is a mobile user $\mathfrak{u}_i$ and the case where object $o$ is a location of interest $\mathfrak{l}_j$.

**Case I** - ($o = \mathfrak{u}_i$): Consider the set of triggers relevant to the mobile user $\mathfrak{u}_i$ denoted by $\mathcal{T}^i$.

$$Pr[S_k(\mathfrak{u}_i(t)) \Rightarrow_{\geq 1} \mathcal{T}^i] = 1 - Pr[S_k(\mathfrak{u}_i(t)) \not\Rightarrow \mathcal{T}^i] \tag{23}$$

Assuming that the probability of activating each trigger in $\mathcal{T}^i$ is independent of the probability of activating any other triggers, we have,

$$Pr[S_k(\mathfrak{u}_i(t)) \not\Rightarrow \mathcal{T}^i] = \prod_{j=1}^{|\mathcal{T}^i|} \Big(1 - Pr[S_k(\mathfrak{u}_i(t)) \Rightarrow \mathfrak{t}_{i,j}]\Big)$$

$$= \prod_{j=1}^{|\mathcal{T}^i|} \Big(1 - Pr[S_1(\mathfrak{l}_j(t')) \in R_{i,j}^1] ... \cdot Pr[S_k(\mathfrak{u}_i(t)) \in R_{i,j}^k]$$

$$... \cdot Pr[S_{f(j)}(\mathfrak{l}_j(t'')) \in R_{i,j}^{f(j)}]\Big)$$

The term $S_{f(j)}$ indicates that the number of streams is a function of the location of interest as it is dependent on the set of attributes being monitored at the location of interest. Given the above equation, we can conclude that the probability of any relevant triggers

being activated by a data update $S_k(\mathfrak{u}_i(t))$ is zero, if (i) $\forall\, j : 1 \leq j \leq \mid \mathcal{T}^i \mid$, $Pr[S_k(\mathfrak{u}_i(t)) \in R^k_{i,j}] = 0$, or (ii) $\forall\, j : 1 \leq j \leq \mid \mathcal{T}^i \mid$, $\exists\, k' : 1 \leq k' \leq f(j) \bigwedge k' \neq k$, $Pr[S_{k'}(\mathfrak{l}_j(t')) \in R^{k'}_{i,j}] = 0$. Note that the time instant $t'$ corresponding to each $k'$ may be different.

**Case II** - ($o = \mathfrak{l}_j$): Consider the set of triggers relevant to the location of interest $\mathfrak{l}_j$ denoted by $\mathcal{T}^j$. We assume that $\mathfrak{l}_j$ has monitored attributes associated with $m$ different data sources.

$$Pr[S_k(\mathfrak{l}_j(t)) \Rightarrow_{\geq 1} \mathcal{T}^j] = 1 - Pr[S_k(\mathfrak{l}_j(t)) \not\Rightarrow \mathcal{T}^j] \tag{24}$$

Again, assuming that the probability of activating each trigger in $\mathcal{T}^j$ is independent of the probability of activating any other triggers, we have,

$$Pr[S_k(\mathfrak{l}_j(t)) \not\Rightarrow \mathcal{T}^j] = \prod_{i=1}^{|\mathcal{T}^j|} \Big(1 - Pr[S_k(\mathfrak{l}_j(t)) \Rightarrow \mathfrak{t}_{i,j}]\Big)$$

$$= \prod_{i=1}^{|\mathcal{T}^j|} \Big(1 - Pr[S_1(\mathfrak{l}_j(t')) \in R^1_{i,j}] \ldots \cdot Pr[S_k(\mathfrak{l}_j(t)) \in R^k_{i,j}]$$

$$\ldots \cdot Pr[S_{k''}(\mathfrak{u}_i(t'')) \in R^{k''}_{i,j}] \ldots \cdot Pr[S_m(\mathfrak{l}_j(t''')) \in R^m_{i,j}]\Big)$$

Given the above equation, we conclude that the probability of any relevant triggers being activated by the data update $S_k(\mathfrak{l}_j(t))$ is zero, if (i) $\forall\, i : 1 \leq i \leq \mid \mathcal{T}^j \mid$, $Pr[S_k(\mathfrak{l}_j(t)) \in R^k_{i,j}] = 0$, or (ii) $\forall\, i : 1 \leq i \leq \mid \mathcal{T}^j \mid$, $\exists\, k' : 1 \leq k' \leq m \bigwedge k' \neq k$, $Pr[S_{k'}(o(t')) \in R^{k'}_{i,j}] = 0$. Note that $o \in \{\mathfrak{u}_i, \mathfrak{l}_j\}$ and the time instant $t'$ corresponding to each $k'$ may be different.

The next section introduces the concept of *safe containment* which allows us to develop data structures modeling fast and precise checking of conditions under which data updates may be dropped.

## *5.4 Safe Containment*

The concept of safe containment can be applied to incoming data updates to ensure that an update with zero probability of activating any relevant triggers may be dropped.

**Definition 5.4.1** ***(Safe Value Container)*** *A safe value container $\psi(o, S_k)$ is defined for each object $o$ with respect to each data source $S_k$ delivering content relevant to the object*

*$o$, such that,*

$$Pr[S_k(o(t)) \Rightarrow_{\geq 1} \mathcal{T}' \mid S_k(o(t)) \in \psi(o, S_k)] = 0 \quad (25)$$

*$\psi(o, S_k)$ is computed as a set of n-dimensional range of values $[L^n, H^n]$, where $L^n = \langle l_1 l_2 ... l_n \rangle$, $H^n = \langle h_1 h_2 ... h_n \rangle$ and $n$ is the dimensionality of the data being delivered by $S_k$.*

**Definition 5.4.2** ***(Inclusive Trigger Set)*** *When the extent of the safe value container $\psi(o, S_k)$ overlaps the predicate range of a set of relevant triggers $\mathcal{T}^I$, $\mathcal{T}^I$ is defined as the inclusive trigger set. Note that any data updates falling within the safe value container can still be dropped as long as the system is aware of the fact that updates for other relevant attributes may activate a trigger $\in \mathcal{T}^I$. As long as the attribute values delivered by $S_k$ fall within the safe value container $\psi(o, S_k)$, any trigger relevant to $o$ and $\notin \mathcal{T}^I$ cannot be activated by updates from other relevant data sources.*

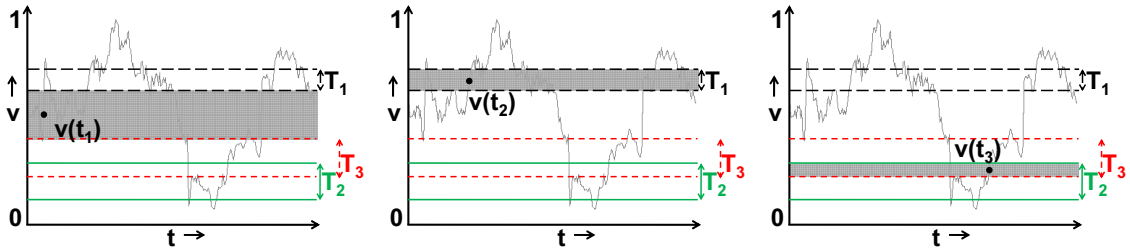
In section 5.4.2, we present two algorithms for safe value container computation for single-dimensional data discussing the pros and cons for each approach. Section 5.4.3 describes an algorithm for safe value container computation for multi-dimensional data. However, before proceeding with the algorithmic details, we establish the properties desired of safe value containers.

### 5.4.1 Safe Value Container Computation Requirements

The goal of safe containment is to minimize the number of data updates which need to be evaluated against the installed triggers in order to determine trigger activation. We identify the following requirements for safe value container computation.

*Lightweight Construction*: Safe value container computation needs to be performed for each object $o$ with respect to attributes delivered by each data source $S_k$ delivering data tuples relevant to the object $o$, thus making it imperative that safe value container computation is performed quickly in order to avoid server overload.

*Fast Containment Check*: The containment check needs to determine if the current attribute




(a) (b) (c)

**Figure 36:** Single-dimensional safe value container computation scenarios. (a) Case I: attribute value lies outside any relevant trigger regions, (b) Case II: attribute value lies inside a relevant trigger region and (c) Case III: attribute value lies within the intersection of multiple relevant trigger regions.

values for object $o$ delivered by data source $S_k$, $S_k(o(t))$, lie within the current safe value container $\psi(o, S_k)$.

*Maximum Coverage*: As long as the data tuple $S_k(o(t))$ lies within the safe value container $\psi(o, S_k)$, the data update can be dropped. Maximizing the *extent* of safe value containers should achieve the goal of minimizing the fraction of data updates processed. However, this may be in direct opposition to the goal of lightweight construction of safe value containers.

### 5.4.2 Single-Dimensional Safe Value Containers

We define single-dimensional safe value containers $\psi(o, S_k)$ for each object $o$ and each data source $S_k$ delivering single-dimensional data relevant to $o$. Safe value containers for single-dimensional data can be represented using a set of range of values $[L^1, H^1]$, such that, as long as the attribute value lies within the bounds of any range of values no triggers relevant to $o$ can be activated. Note that triggers in the inclusive trigger set may be activated by updates from other data sources. Figure 36 displays the plot of a single-dimensional attribute $v$ varying with time $t$. The predicate conditions for three different installed triggers $T_1$, $T_2$ and $T_3$ over the attribute $v$ are displayed by horizontal lines. We identify three separate cases for safe value container computation.

**Case I:** When attribute value lies outside any relevant trigger range values, the safe value container is identified as the range $[l, h]$, such that, no relevant triggers can be activated

as long as the attribute value lies within the safe value container. Figure 36(a) shows an example where the value of the attribute at time $t_1$, $v(t_1)$, lies in a safe value container, $[l, h] = [T_3.high, T_1.low]$. This safe value container remains valid till the attribute value lies between $T_3.high$ and $T_1.low$. As soon as the attribute value $v(t) < T_3.high$ or $v(t) > T_1.low$, the safe value container is invalidated and recomputed.

**Case II:** When attribute value lies within a relevant trigger range, the safe value container is identified as the range $[l, h]$, such that $l, h$ represent the lowerbound and the upperbound of the predicate condition associated with the trigger. As long as the attribute value lies within the safe value container no triggers can be activated. However, updates to other relevant attributes may result in activation of the *inclusive trigger*. Figure 36(b) shows an example scenario where the attribute value $v(t_2)$ lies within the trigger range of $T_1$ and the safe value container is identified as $[l, h] = [T_1.low, T_1.high]$. Trigger $T_1$ is an inclusive trigger in this scenario.

**Case III:** When attribute value lies within multiple relevant intersecting trigger ranges, the safe value container is identified as the *minimal intersection* of the trigger value ranges. The intersecting triggers form the *inclusive trigger set*. Figure 36(c) shows an example scenario where the attribute value $v(t_3)$ lies within the trigger range of $T_2$ as well as $T_3$. The safe value container is identified as $[l, h] = [T_3.low, T_2.high]$. Triggers $T_2$ and $T_3$ form the inclusive trigger set. Once $v(t)$ moves outside the bound $[l, h]$, safe value container recomputation is required.

Multiple safe value container computation is more expensive compared to single safe value container computation. However, *multiple safe value containers lead to fewer data updates compared to single safe value containers when the rate of change of data values is high and update frequency is low*. We now discuss algorithms for single and multiple safe value container computation. The advantage of multiple safe value containers over single safe value containers is depicted with the help of an example.

#### 5.4.2.1 Single Safe Value Container Computation Algorithm

Algorithm 3 outlines the procedure for single safe value container computation. The algorithm presented here computes a single safe value container in accordance with the three case scenarios described above. The algorithm accepts as input the object $o$ and the current value $a$ of the single-dimensional attribute and outputs the safe value container $[l, h]$ associated with the object $o$ dependent on the current value of the attribute.

**Algorithm 3**: Single Safe Value Container Computation Algorithm

```
   Input: o, a
   Output: [l, h]
 1 [r_l, r_h] ← getRangeBlock(a);
 2 T' ← getIntersectingTriggers(o, [r_l, r_h]);
 3 l ← r_l, h ← r_h;
 4 if (T' ≠ φ) then
 5     for (i = 0; i < |T'|; i++) do
 6         if (T'_i.low <= a && T'_i.low > l) then
 7             l ← T'_i.low;
 8         end
 9         else if (T'_i.low >= a && T'_i.low < h) then
10             h ← T'_i.low;
11         end
12         if (T'_i.high >= a && T'_i.high < h) then
13             h ← T'_i.high;
14         end
15         else if (T'_i.high <= a && T'_i.high > l) then
16             l ← T'_i.high;
17         end
18     end
19 end
20 return [l, h];
```

In order to limit the safe value container computation costs, we consider a small number of triggers in the vicinity of the current attribute value $a$. To facilitate this, the value range of the attribute is divided into a number of blocks and the range block $[r_l, r_h]$ within which the current attribute value $a$ lies is computed (line 1). Next, we retrieve the set of triggers $\mathcal{T}'$ relevant to object $o$ intersecting the range block $[r_l, r_h]$ (line 2). If the set of intersecting triggers $\mathcal{T}'$ is empty, the entire range block is returned as the safe value container. Otherwise, the algorithm proceeds to compute the safe value container dependent on the intersecting trigger bounds (lines 4-19).

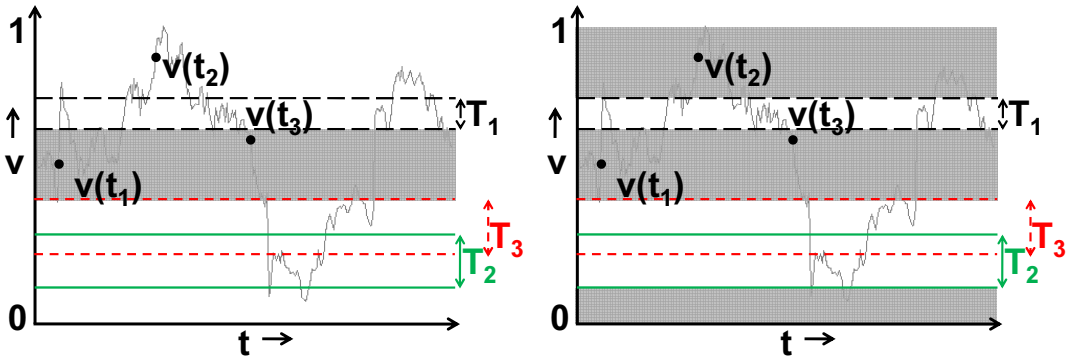



**Figure 37:** Single Safe Value Container vs. Multiple Safe Value Containers. (a) Data source delivers data with high rate of change and low update frequency invalidating the single safe value container at time instant $t_2$. (b) Multiple safe value containers avoid the unnecessary recomputation of safe value containers at time instants $t_2$ and $t_3$.

For each of the intersecting triggers, the algorithm considers the bounds $\mathcal{T}'_i.low$ and $\mathcal{T}'_i.high$ associated with the trigger. The lower and upper bounds of the safe value container are modified to the nearest trigger bounds around the current attribute value as outlined in the algorithm (lines 6-17).

#### *5.4.2.2 Multiple Safe Value Container Computation Algorithm*

Multiple safe value containers are computationally more expensive than single safe value containers. However, in certain scenarios the disadvantages associated with the higher computational costs are outweighed by the advantages offered by multiple safe value containers in terms of lower trigger processing costs.

Figure 37 displays an example scenario where the source generates data updates $v(t_1)$, $v(t_2)$ and $v(t_3)$ at time instants $t_1$, $t_2$ and $t_3$. Consider the single safe value container in Figure 37(a). The data source being considered here delivers data which has a *high rate of change* and *low update frequency*. The value $v(t_2)$ at time instant $t_2$ lies outside the current safe value container. Hence, the safe value container is invalidated and a new safe value container is computed. At time instant $t_3$, the data value again lies within the safe value container associated with $v(t_1)$; as this safe value container has been previously invalidated

on receiving data update $v(t_2)$, the safe value container associated with $v(t_1)$ is recomputed as the new safe value container at time instant $t_3$.

Figure 37(b) displays multiple safe value containers, denoted by shaded stripes, which avoid the unnecessary recomputation of safe value containers described in the above scenario. The values $v(t_1)$, $v(t_2)$ and $v(t_3)$ lie inside one of the multiple safe value containers at each time instant, thus preventing data updates which the single safe value container approach required.

**Algorithm 4**: Multiple Safe Value Container Computation Algorithm

```
   Input: o, a
   Output: {svcBlocks}
 1 [r_l, r_h] ← getRangeBlock(a);
 2 T' ← getIntersectingTriggers(o, [r_l, r_h]);
 3 {svcBlocks} ← [r_l, r_h];
 4 if (T' ≠ φ) then
 5 |  getMultSvc ← true;
 6 |  for (i = 0; i < |T'|; i++) do
 7 |  |  if (T'_i.low <= a && a <= T'_i.high) then
 8 |  |  |  getMultSvc ← false;
 9 |  |  |  break;
10 |  |  end
11 |  end
12 |  if (!getMultSvc) then
13 |  |  {svcBlocks} ← φ;
14 |  |  [l, h] ← compSingleSvc(o, a);
15 |  |  {svcBlocks} ← [l, h];
16 |  end
17 |  else
18 |  |  {svcBlocks} ← [r_l, r_h] - ⋃_{i=1}^{|T'|} [T'_i.low, T'_i.high];
19 |  end
20 end
21 return {svcBlocks};
```

Algorithm 4 outlines the basic algorithm for computing multiple safe value containers. The algorithm accepts as inputs the object $o$ and the attribute value $a$ associated with the object $o$ and delivered by the single-dimensional data source. A set of multiple safe value containers $\{svcBlocks\}$ is computed. Similar to algorithm 3, the value range of the attribute is divided into a number of range blocks. The range block $[r_l, r_h]$ within which the current attribute value $a$ lies is computed (line 1) and the set of triggers $\mathcal{T}'$ relevant to

object $o$ intersecting the range block $[r_l, r_h]$ is computed (line 2). If the set of intersecting triggers is empty, the range block is returned as the set $\{svcBlocks\}$. Otherwise, the algorithm determines if any of the intersecting trigger regions contain the current attribute value $a$ (lines 5-11). If the attribute value $a$ lies within any relevant trigger regions, the algorithm resorts to computing a single safe value container using the procedure outlined in algorithm 3 (lines 12-16). In case the attribute value lies outside all relevant trigger regions, the algorithm computes multiple safe value containers as illustrated in Figure 37(b). The multiple safe value containers represent the range block except the region covered by the union of relevant trigger ranges $[r_l, r_h]$ - $\bigcup_{i=1}^{|\mathcal{T}'|} [\mathcal{T}'_i.low, \mathcal{T}'_i.high]$ (lines 17-19).

### 5.4.3 Multi-dimensional Safe Value Containers

In the previous section, we described algorithms for computation of safe value containers for single-dimensional data. However, data streams may deliver multi-dimensional data instead of single-dimensional data, thus requiring computation of multi-dimensional safe value containers. The attributes being delivered by a multi-dimensional data source are not independent of each other; thus, considering these attributes to be independent and computing single-dimensional safe value containers for each dimension leads to inefficient data processing. For example, the data stream delivering location updates provides two-dimensional data $(x, y)$ denoting the current location of the mobile user.

Similar to single-dimensional safe value containers, multi-dimensional safe value containers aim to reduce the number of data updates which need to be evaluated against the set of installed triggers. Section 5.4.1 defined the requirements for safe value container characteristics. Multi-dimensional safe value containers may have some additional requirements like *compact representation*. For example, two-dimensional safe value containers for location updates need to be communicated back to the mobile user over a wireless channel which may lead to significant communication costs and energy consumption on the mobile client. A rectangular-shaped two-dimensional safe value container requires only two points

(bottom-left and top-right) for representation and can be considered to be suitable to meet the requirement of compact representation. More complex-shaped (irregular) safe value containers may have higher overhead for communication due to more complex representation.

#### *5.4.3.1 Two-dimensional Safe Value Container Computation Algorithm*

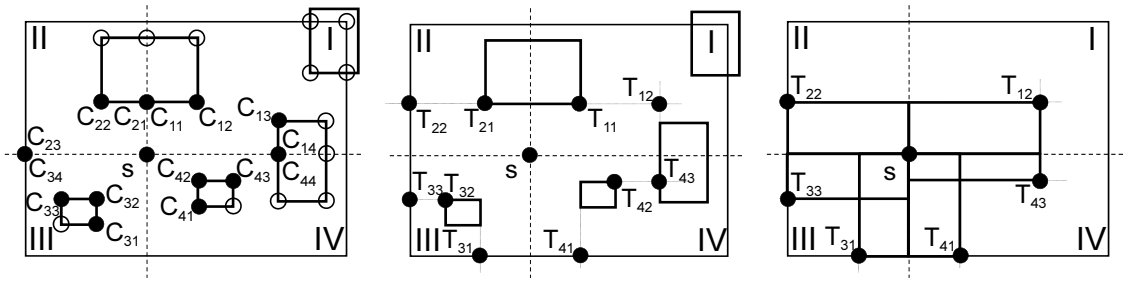
We now outline the procedure for computation of two-dimensional safe value containers. The underlying concepts may be applied to extend this approach to safe value container computation for higher dimensions. We use the concept of dominating point and appropriate heuristics to find the skyline points [88, 101] which form the corner points of the rectangular safe value container in two-dimensional space. In order to reduce the safe value container computation costs, relevant triggers in the data space only in the vicinity of the current value $(a_1, a_2)$ are considered. This is achieved by overlaying a grid structure over the two-dimensional space associated with the data being delivered by the source. The algorithm accepts the two-dimensional data value $(a_1, a_2)$ and the current two-dimensional grid cell in which the data value resides as inputs. The set of triggers intersecting this two-dimensional grid cell are considered for safe value container computation. This step is analogous to *range block* computation for single-dimensional data. In case no relevant triggers intersect the grid cell, the entire two-dimensional cell is returned as the safe value container. Otherwise, the algorithm proceeds to calculate the safe value container by applying the concept of dominating points and skyline computation as mentioned earlier.

The algorithm partitions the two-dimensional grid cell into four quadrants with $(a_1, a_2)$ as the origin. We define a set of *candidate points* and a set of *tension points* for each quadrant. The candidate point set is the set of points which can potentially form a corner point of the rectangular safe value container. Tension points are obtained from the set of candidate points by ensuring that only points that form largest possible rectangular regions not overlapping the spatial region associated with any relevant trigger are selected.

The set of candidate points is determined as follows. Firstly, the spatial region corner for each relevant trigger is selected as a candidate point in its appropriate quadrant. For triggers which do not completely lie inside the grid cell, the intersection points of the cell boundary and the trigger spatial region are considered as candidate points instead of the corner points which fall outside the grid cell. Secondly, for trigger spatial regions which intersect the x-axis or y-axis of the coordinate axes with origin at $(a_1, a_2)$, we also consider points of intersection of the triggers with the axes as candidate points. The algorithm trims the set of candidate points in the next step. Firstly, in case multiple candidate points in a quadrant intersect the x-axis (or y-axis), all candidate points other than the point on the x-axis (or y-axis) closest to the origin are removed from the candidate point set. If no intersecting points are present on the x-axis, the point of intersection of the x-axis and the cell is added to the candidate point set. Further, we remove points which *fully dominate* any other point from the candidate set. A point $P_1$ is said to fully dominate point $P_2$ if $P_1.x > P_2.x$ and $P_1.y > P_2.y$. Finally, the points are sorted according to increasing distance of the x-coordinate from the origin. Points with the same x-coordinate are arranged in order of decreasing distance of y-coordinate from origin.

The set of candidate points is then processed in the following manner to obtain the set of tension points. Each tension point $T_{Qi}$, where $Q \in \{1, 2, 3, 4\}$ represents the quadrant the point belongs to, is assigned the same x-coordinate as the corresponding candidate point $C_{Qi}$. $T_{Qi}$ is assigned the same y-coordinate as that of $C_{Qi-1}$, or $T_{Qi-1}$ if $T_{Qi}$ and $T_{Qi-1}$ have the same x-coordinate. The y-coordinate of $T_{Q1}$ is set as either the top bound of the cell or the y-coordinate of a candidate point intersecting the y-axis if any.

The set of tension points form the opposite corner (opposite to the origin) of the set of candidate *component rectangles* in each quadrant. The final safe value container is composed of the intersection of the component rectangles from each quadrant. Various heuristics may be applied to determine the set of component rectangles to be used for composing the rectangular safe value container.




(a) Candidate Point Set (b) Tension Point Set (c) Component Rectangles

**Figure 38:** Two-dimensional Safe Value Container Computation

We now describe a simple greedy heuristic in order to quickly compose a suitable safe value container from the set of component rectangles. As opposed to an optimal solution which enumerates every possible combination of component rectangles and computes metrics for each combination thus taking quartic time, our approach performs greedy decisions. For example, we may attempt to determine the rectangular safe value container with the largest perimeter in order to reduce the cost of trigger evaluations. To compute this safe value container, a simple approach can be to first select the quadrant in which the component rectangle with the largest perimeter is present. Quadrants are further selected dependent on the perimeter of the largest component rectangle present in the quadrant. At each step, the component rectangle which forms the rectangular safe value container with the largest perimeter is selected. The algorithm continues until all four quadrants are processed using this greedy heuristic.

Figure 38 shows an example of our safe value container computation approach. The candidate point set for the given scenario is as shown in Figure 38(a). The black dots represent the candidate points, whereas the hollow dots represent points which are trimmed from the candidate point set as explained above. Figure 38(b) displays the set of tension points obtained from the candidate point set. Figure 38(c) displays the component rectangles formed by selecting a few of the tension points. The final rectangular safe value container is composed out of these component rectangles using the heuristic described above.

Extension of this approach to larger dimensional spaces may lead to expensive computations. For example, three-dimensional spaces would require the processing of relevant points in each octant of the coordinate system. Our experimental evaluation is limited to single-dimensional and two-dimensional spaces.

## *5.5 Experimental Evaluation*

In this section, we evaluate the performance of the SLIM system to exhibit the benefits of our safe containment techniques. We benchmark the performance of the SLIM approach against three different approaches: in-transit processing (*INT*) which processes all data updates in the order of their arrival (as outlined in section 5.1), random update dropping (*RND*) and the safe value container approach applied to the spatial attribute (*SR*) alone. The experiments are designed to study the performance of these techniques based on the following metrics: 1) *Computation costs*. These are measured as a combination of trigger processing costs as well as safe value container computation costs. Lower computation costs are desired in order to enhance system scalability. 2) *Communication costs and energy consumption*. The system aims at reducing client-to-server communication costs, especially in a wireless environment where high communication costs also lead to excessive energy consumption on the mobile client. 3) *Trigger activation success rate*. We aim to achieve 100% trigger activation success rate in order to provide a high level of QoS. The sequence of triggers which should be activated are determined by the in-transit processing approach. This approach has very high computation costs, thus, it is practically impossible for this approach to activate all triggers. We relax the real-time processing requirements for this approach in order to determine the trigger activation sequence.

### 5.5.1 Experimental Setup

We simulate the proposed mobile information monitoring system using an event-based simulator. The event-based simulator consists of two parts, one responsible for generating mobile user movement traces and the other responsible for modeling the arrival of data

from multiple sources at a large number of locations of interest. The mobile user movement simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS [8]) in Spatial Data Transfer Format (SDTS [10]). Vehicles are randomly placed on the road network, according to traffic densities determined from the traffic volume data in [53], ensuring appropriate traffic densities on different road classes like expressway, collector and arterial roads.

The simulator simulates the motion of vehicles on the road network with the velocity information presented in Table 3. At the intersections, vehicles may move in any direction with attached probability values in order to keep traffic densities on various road classes constant. We use a map of Atlanta and surrounding regions, which covers an area around 1000 $km^2$ in expanse, to generate the trace. Our experiments use traces generated by simulating vehicle movement for a period of one hour with results averaged over a number of such traces. Traffic volume values allow us to simulate the movement of a set of 10,000 vehicles with each vehicle generating a set of position parameters during the simulation. Default values require each vehicle to generate location updates with a period of $0.5$ seconds.

The other part of the simulator models multiple data sources generating data relevant to the locations of interest. A default set of 10,000 locations of interest uniformly distributed over the entire map region are considered. The data arrival at the mobile information monitoring server is modeled as a poisson process; hence, the interarrival times are exponentially distributed with the mean arrival rate as expressed in Table 3. A maximum of twenty data sources are considered in the system with a different set of attributes monitored at each location of interest. The default trigger information consists of a set of 10,000 triggers with the fraction of private, shared and public triggers installed in the system determining the number of triggers relevant to each client. We consider a dynamically changing trigger set where triggers are inserted and deleted periodically.

**Table 3:** Experimental Setup Parameters

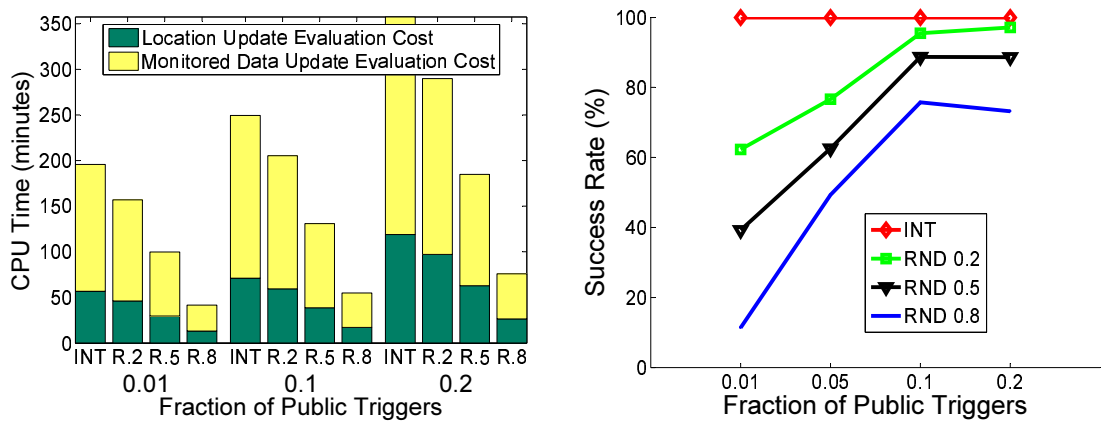
| Parameter | Default Value | Value Range |
|---|---|---|
| # Mobile Users | 10K | – |
| # Locations of Interest | 10K | – |
| # Triggers | 10K | – |
| # Data Sources | 10 | 10 - 20 |
| Grid Cell Size (sqkm) | 1.1 | 0.01 - 10 |
| Range Block (% of value range) | 50% | 5%-50% |
| Location Update Period | 0.5 sec | – |
| Data Update Period | 1, 2, 5, 10 sec | 1 - 20 sec |
| Mean User Speed (km/h) | {50, 60, 90} | – |
| User Speed Std. Dev. (km/h) | {10, 15, 20} | – |
| Data Value Rate of Change Mean (per sec) | {0.0625%, 0.125%, 0.25%, 0.5%, 1%} | – |
| Data Value Rate of Change Std. Dev. (per sec) | {0.005%, 0.01%, 0.02%, 0.04%, 0.08%} | – |
| Fraction of *Private*, *Shared* and *Public* Triggers | {0.6, 0.3, 0.1} | {0.45-0.66, 0.3-0.35, 0.01-0.2} |

Each experimental setup considers default values, as shown in Table 3, unless mentioned otherwise. This simulator setup allows us to test the robustness of our framework under realistic mobility and non-spatial data arrival patterns. Indexing structures like B-tree [37], R*-tree [31] and X-tree [33] are used for single-dimensional, two-dimensional and multi-dimensional data respectively. Even installed triggers are indexed using these indexing structures. Note that the performance gain of our safe containment techniques is independent of the underlying indexing mechanisms.

### 5.5.2 Experimental Results

#### *5.5.2.1 Limitations of In-transit Processing and Random Dropping Approaches*

This experiment is designed to expose the limitations of the in-transit processing and random dropping approaches. We vary the fraction of public triggers from 0.01 to 0.2 - higher fraction of public triggers results in each client having more relevant triggers in the system - and study the performance of the in-transit processing approach (INT) and the random dropping approach for drop probabilities of 0.2, 0.5 and 0.8 (RND 0.2, RND 0.5, RND

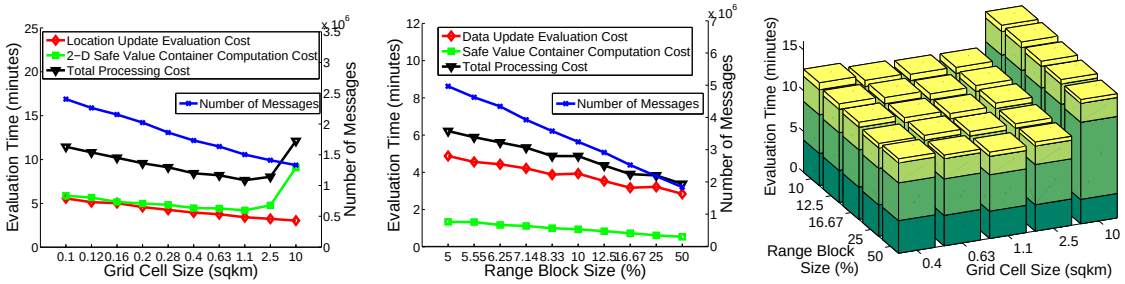



(a) (b)

**Figure 39:** Performance of in-transit processing and random dropping with varying fraction of public triggers

0.8).

Figure 39(a) plots the CPU time for each approach as we vary the fraction of public triggers. It can be observed from the figure that the in-transit approach is not at all scalable even for the lowest fraction of public triggers as it requires nearly 200 minutes of CPU time to process data received over 60 minutes (one hour trace). Figure 39(b) displays the success rate for each approach and shows that this approach has 100% success rate. However, for all practical purposes, due to high computational costs, this approach will start dropping updates due to the heavy load on the server. A simple alternative is to randomly drop updates as they are received at the server. This approach suffers from two deficiencies. Figure 39(a) shows the CPU time required for random dropping approaches with different drop probabilities. The CPU load reduces as we increase the drop probability for the random dropping approach. However, even with very high drop probability of 0.8 the system is not scalable for higher fraction of public triggers. Higher processing costs are incurred as we increase the fraction of public triggers installed in the system. Hence, it is not possible to determine the ideal drop probability to ensure system scalability; alternatively, one can vary the drop probability with CPU load. However, the random dropping approaches reduce CPU load at the cost of drop in success rate as can be observed from Figure 39(b).




(a) (b) (c)

**Figure 40:** (a) Results with varying grid cell size for two-dimensional safe value containers. (b) Results with varying range block size for single-dimensional safe value containers. (c) Results with varying grid cell size and range block size for single-dimensional and two-dimensional safe value containers.

For low fraction of public triggers this success rate provides unacceptable QoS. Safe value container-based approaches rectify the situation by allowing the system to determine and drop only the updates which have zero probability of activating relevant triggers.

#### *5.5.2.2 Determining Range Block Size and Grid Cell Size*

This experiment is aimed at determining the appropriate grid cell size for two-dimensional safe value containers and the range block size for single-dimensional safe value containers. First, we consider spatial and non-spatial data streams being delivered separately to the server. Finally, we consider a mobile information monitoring system which receives both spatial and non-spatial data streams concurrently. Figure 40(a) displays the evaluation time and the number of update messages received at the server for a location data stream utilizing the two-dimensional safe value container computation algorithm explained in section 5.4.3.1. The figure shows the results for evaluation time as we vary the grid cell size from 0.1 $km^2$ to 10 $km^2$, plotted on the left y-axis. There are two costs involved in the system: two-dimensional safe value container computation cost and the location update evaluation cost for mobile clients positioned outside their current safe value container.

We can observe from the figure that the safe value container computation cost first slowly decreases as we increase the grid cell size. This is due to lower number of safe value container computations being performed for larger grid cell sizes. As the grid cell size is

increased, a larger number of triggers in the vicinity of the user are considered, allowing for computation of larger safe value containers. This leads to lower number of messages being processed at the server for larger grid cell sizes as mobile clients stay within the safe value container for longer time periods. The number of messages processed are plotted on the right y-axis. However, with increasing grid cell sizes, each safe value container computation becomes more expensive as it considers a larger number of trigger regions. For the largest grid cell size of 10 $km^2$, each safe value container computation is expensive enough to outweigh the effect of lower number of safe value container computations being performed. Thus, the safe value container computation costs exhibit an increase for this grid cell size. The location update evaluation costs decline with increasing grid cell sizes as fewer number of messages are evaluated at the server. The overall computation costs are lowest for a grid cell size of 1.1 $km^2$ as visible from the figure.

Similarly, Figure 40(b) plots the evaluation times and the number of messages as we vary the range block size for single-dimensional data on the left y-axis. The safe value container computation costs steadily decline as we increase the range block size. Again, this is due to fewer number of messages being processed at the server with increasing range block size, as plotted on the right y-axis. Larger range block sizes allow us to compute safe value containers of larger extent which leads to fewer number of data updates being processed at the server. Even though the cost of each safe value container computation increases with increasing range block size, the overall safe value container computation costs slowly decline. Data update evaluation costs also decrease with increasing range block size. Range block size of 50% has lowest total processing costs.

Figure 40(c) displays the evaluation time for the SLIM system with varying grid cell size for two-dimensional location updates and range block size for single-dimensional data updates. The figure plots the location update evaluation cost, two-dimensional safe value container computation cost, data update evaluation cost and the single-dimensional safe value container computation cost as stacked bars for different combinations of grid cell

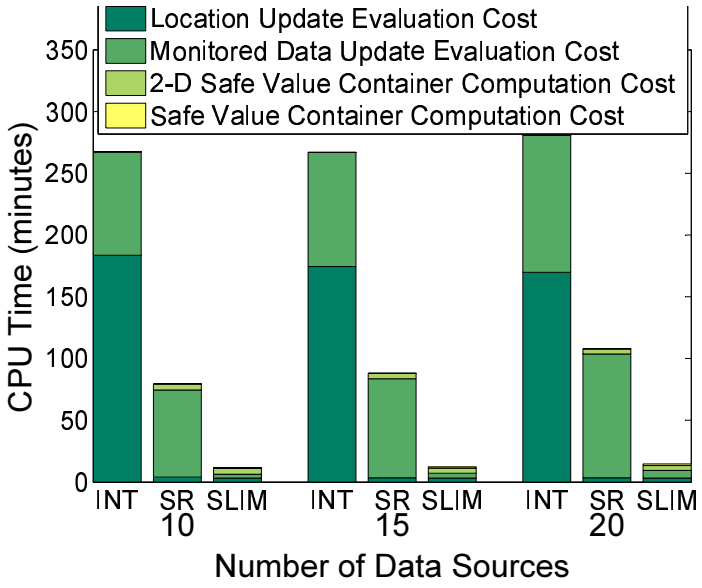



**Figure 41:** Performance comparison of in-transit, spatial safe value container and SLIM

size and range block size. SLIM also displays lowest overall evaluation costs for the combination of grid cell size of 1.1 $km^2$ and range block size of 50%. We use these parameters for further experimentation.

#### *5.5.2.3 System Scalability*

Our mobile system is designed with the goal of achieving 100% trigger activation success rate. The following experiment considers only those approaches which can meet this criteria and studies the performance of SLIM in comparison with in-transit processing (INT) and the safe value container approach applied to spatial data alone (SR). Note that the success rate is not plotted as all of the above approaches achieve 100% trigger activation success rate.

As can be seen from Figure 41, the SLIM approach outperforms in-transit processing as well as the SR approach. The SLIM approach shows that interaction of spatial and non-spatial data attributes further enhances system scalability. SLIM outperforms in-transit processing by a margin of around 20 times and the SR approach by a margin of 5-8 times. Updates which may satisfy their respective predicate conditions can still be dropped, in

case, other predicate conditions are not likely to be satisfied for trigger evaluation. Due to this interaction between attributes from different data sources, the SLIM approach exhibits scalability even as we increase the number of data sources. The other two approaches fail to exploit the effects of this interaction between multiple attributes.

#### *5.5.2.4 Energy and Bandwidth Consumption at Client*

Figure 42(a) displays the bandwidth consumption for mobile clients as we vary the fraction of public triggers from 0.01 to 0.2. The bandwidth consumption is measured in terms of number of messages sent from the mobile clients to the server plotted on a logarithmic scale. As can be seen from the figure, the in-transit processing approach requires a large number of client-to-server messages. However, the number of messages remains constant as we vary the fraction of public triggers. SLIM requires only 1.5% - 3.6% of number of client-to-server messages required by the in-transit processing approach. The SR approach requires same number of client-to-server messages as SLIM. This is due to the fact that each mobile client in the SR approach is also aware of its own safe value container as in SLIM. The benefit of SLIM over SR approach is experienced only in terms of enhanced server scalability where the server may drop a position update from the mobile client based on the interactions with other attributes. The mobile client cannot determine any such interactions with other attributes without communicating with the server. Another observation from the figure is that the number of client-to-server messages increases as we increase the fraction of public triggers. At higher fraction of public triggers, higher number of triggers are relevant to each client thus resulting in smaller safe value containers. As a result, mobile clients move out of their safe value containers more often and need to communicate with the server more frequently.

Figure 42(b) displays the energy consumption on each mobile client in milliwatt-hours. Once again, SLIM and SR approaches have the same energy consumption on the mobile

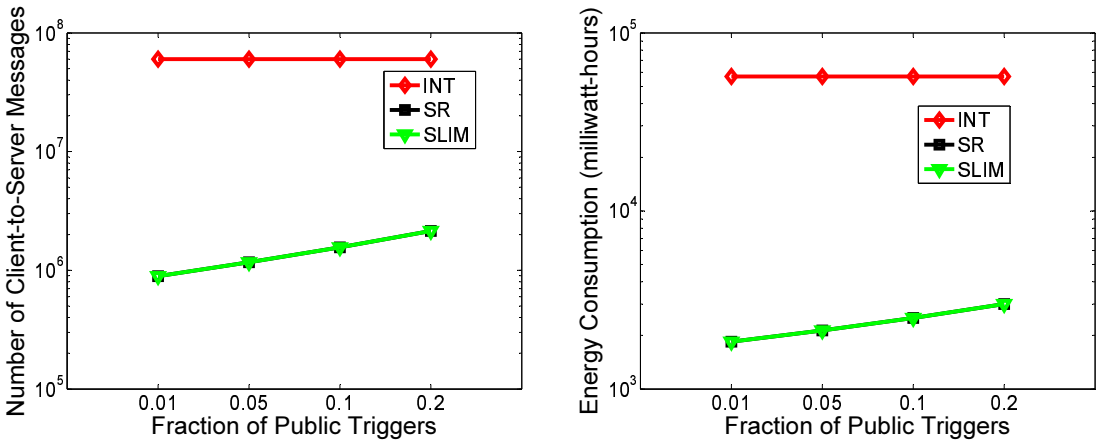



**Figure 42:** Performance comparison of bandwidth and energy consumption at mobile client.

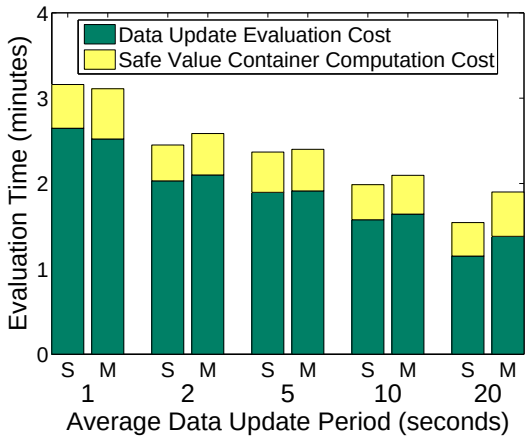
client. As we vary the fraction of public triggers, the energy consumption of these approaches varies from 2.8% - 4.6% of the energy consumption on a mobile client under the in-transit processing approach. The in-transit approach has the same energy consumption irrespective of the fraction of public triggers. Energy consumption is estimated as a combination of power required for client-to-server communication and the computation required on the client [107]. Wireless communication costs form the dominant part of energy consumption costs on mobile clients. Lower communication costs provide a huge benefit in terms of lower energy consumption on the mobile client. SLIM also requires the mobile client to monitor its position within the safe value container, thus leading to some additional energy consumption. However, the energy consumption due to the computation on the client is negligible when compared to the savings in energy consumption due to lower wireless communication costs.

#### *5.5.2.5 Single (S) vs. Multiple (M) Safe Value Containers*

This set of experiments provides a comparison of the performance of the two approaches for safe value container computation for single-dimensional data, namely, single and multiple safe value container computation. Figure 43 displays the results as we vary the average

| Average Update Period (seconds) | Single SVC (# messages in millions) | Multiple SVC (# messages in millions) | Gap (%) |
|---|---|---|---|
| 1 | 1.749 | 1.749 | 0 |
| 2 | 1.643 | 1.640 | 0.18 |
| 5 | 1.666 | 1.638 | 1.56 |
| 10 | 1.622 | 1.558 | 3.95 |
| 20 | 1.490 | 1.403 | 5.84 |

(a)




(b)

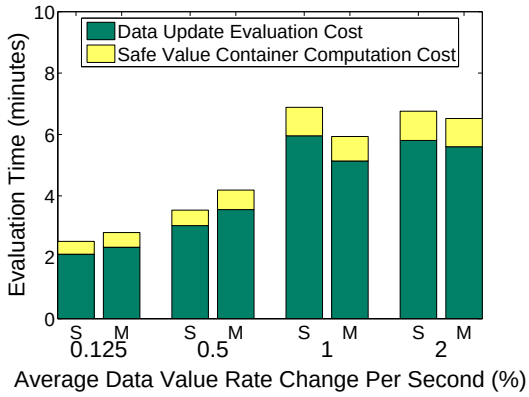
**Figure 43:** Performance comparison of single vs. multiple safe value containers with varying update period.

update period for the data streams from 1 second to 20 seconds. Figure 43(a) displays the number of messages processed by each approach as we increase the update period. At high data frequency (low update period), the gap between the single and multiple approaches is nominal and they process almost the same number of updates. However, as we increase the update period, a significant gap appears between the number of updates processed by the single and multiple safe value container approaches. This is due to the data values *jumping between* the multiple safe value containers leading to significantly larger number of dropped updates. Figure 43(b) displays the evaluation times for each approach as we vary the average update period. Note that the multiple safe value container approach does not perform significantly worse than the single safe value container approach, although the container computation times for this approach are a little higher.

Figure 44 displays the results as we vary the average rate of change of data values from 0.125% to 2% per second. Figure 44(a) displays the number of messages processed by each approach as we increase the average rate of change of data values. At low rate of change of data, the gap between the single and multiple approaches is almost invisible. However, as we increase the data change rate, a significant gap appears between the number of updates processed by the two approaches. Figure 44(b) displays the evaluation times for

| Average Data Value Rate Change Per Second (%) | Single SVC (# messages in millions) | Multiple SVC (# messages in millions) | Gap (%) |
|---|---|---|---|
| 0.125 | 1.544 | 1.542 | 0.13 |
| 0.5 | 1.998 | 1.953 | 2.25 |
| 1 | 2.654 | 2.516 | 5.2 |
| 2 | 3.312 | 2.888 | 12.8 |

(a)




(b)

**Figure 44:** Performance comparison of single vs. multiple safe value containers with varying rate of change of data.

each approach as we vary the average rate of change of data. The total evaluation time for the multiple container approach is slightly lower than that for the single container approach at high rate of change of data due to significantly lower number of processed updates.

## *5.6 Related Work*

An event-based location reminder system has been advocated by many human computer interaction projects [78, 97, 40, 80, 66]. In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand [76, 30]. The SLIM system also needs to deal with the complexity of monitoring continuously changing data from multiple data sources in order to trigger relevant alerts in a non-intrusive manner.

In the field of location-based queries, periodic reevaluation approach is commonly used for continuous monitoring of moving objects [60, 83, 93, 115]. Incremental reevaluation for range and kNN queries has been proposed in [83, 113]. [114, 116] propose the idea of returning moving query results with a validity scope. The SLIM system differs from this work in two aspects. Firstly, our installed location-centric triggers do not demand periodic evaluation or reevaluation like continuous queries; instead they require one-shot

evaluation which should result in a notification being sent to the user when the trigger activation conditions are satisfied. Our work is focused on determining the opportune moment for evaluating triggers relevant to a client. Secondly, unlike the SLIM system, none of the previous work explores the impact of considering both spatial as well as non-spatial predicates and the interaction between the different attributes for information processing in mobile systems.

Streaming data processing optimizations include batch processing methods [22, 75] or approximation processing [39, 106] to handle the large amounts of data. Fortunately, our problem defines a finite number of objects, mobile users and locations of interest, for which an infinite amount of continuously evolving information is being delivered. This makes it possible to store data corresponding to each object in the form of its safe value container. Data updates are processed against safe value containers to determine if installed triggers may be activated by an incoming update.

# CHAPTER VI

# PRIVACYGRID ARCHITECTURE FOR ANONYMOUS LOCATION-BASED SERVICES

## *6.1 Introduction*

With rapid advances in mobile communication technologies and continued price reduction of location tracking devices, location-based services (LBSs) are widely recognized as an important feature of the future computing environment [38]. Though LBSs provide many new opportunities, the ability to locate mobile users also presents new threats – the intrusion of location privacy [34, 53]. According to [34], location privacy is defined as the ability to prevent unauthorized parties from learning one's current or past location.

In LBSs, there are conceivably two types of location privacy – personal subscriber-level privacy and corporate enterprise-level privacy. Personal subscriber-level privacy must supply rights and options to individuals to control when, why and how their location is used by an application. With personal subscriber-level privacy, each individual has liberties to "opt in" and "opt out" of services that take advantage of their mobile location. Corporate enterprise-level privacy is fundamentally different in that corporate IT managers typically control when, why and how mobile location capabilities provide application benefits to the organization as a whole. Within the enterprise, personal subscriber-level privacy is sometimes irrelevant because location is a critical requirement for staff to function productively while on the road. Asset tracking and workforce management are examples of location-enabled enterprise applications. However, companies need enterprise-level privacy to preserve corporate secrets and maintain competitive edge. Location privacy threats refer to the risk that an adversary can obtain unauthorized access to raw location data, derived or computed location information by locating a transmitting device, hijacking the

location transmission channel and identifying the subject using the device [54]. For example, location information can be used to spam users with unwanted advertisements or to learn about users' medical conditions, unpopular political or religious views. Inferences can be drawn from visits to clinics, doctor's offices, entertainment clubs or political events. Public location information can lead to physical harm, such as stalking or domestic abuse.

Several approaches have been proposed for protecting the location privacy of a user. We classify these techniques into three categories: (1) Location protection through user-defined or system-supplied privacy policies; (2) Location protection through anonymous usage of information; and (3) Location protection through pseudonymity of user identities, which uses an internal pseudonym rather than the user's actual identity. As described in [34], some location-based services can operate completely anonymously, such as "*when I pass a gas station, alert me with the unit price of the gas*". Others can not work without the user's identity, such as "*when I am inside the office building, let my colleagues find out where I am*". Between these two extremes are those applications that cannot be accessed anonymously but do not require the user's true identity, such as "*when I walk past a computer screen, let me teleport my desktop to it*". Here, the application must know whose desktop to teleport but it could do this using an internal pseudonym rather than the user's true identity.

For those LBSs that require true user identity, strong security mechanisms such as location authentication and authorization have to be enforced in conjunction with their location privacy policy. In this chapter, we concentrate on the class of location-based applications that accept pseudonyms and present the PRIVACYGRID framework for performing personalized anonymization of location information through customizable location $k$-anonymity and location $l$-diversity, thus enabling anonymous location-based queries in mobile information delivery systems.

Perfect privacy is clearly impossible as long as communication takes place. An important question here is *how much privacy protection is necessary?* Moreover, users often have

varying privacy needs in different contexts. In PRIVACYGRID, we propose to use location $k$-anonymity and location $l$-diversity as two quantitative metrics to model the location privacy requirements of a mobile user. In the context of LBSs and mobile users, location $k$-anonymity refers to $k$-anonymous usage of location information. A user is considered location $k$-anonymous if and only if the location information sent from the mobile user to a LBS is indistinguishable from the location information of at least $k-1$ other users. Location $l$-diversity is introduced to strengthen the privacy protection of location $k$-anonymity in situations where location information shared by the $k$ users is sensitive. Increasing $l$ value to two or higher significantly reduces the probability of linking a static location or a symbolic address (such as church, restaurant, doctor's office) to a mobile user. Location perturbation is an effective technique for implementing *personalized* location $k$-anonymity and location $l$-diversity. Cloaking methods typically perturb the location information by reducing its resolution in terms of time and space, referred to as spatial cloaking and temporal cloaking respectively [53]. By reducing the spatial resolution, a spatial region that contains $k-1$ other users' location information can be used to replace the spatial position of the user. By reducing the temporal resolution, the message can be delayed for a certain duration of time; the message may live long enough to find a spatio-temporal cloaking region that satisfies the maximum spatial and temporal resolution constraints and is shared by at least $k-1$ other mobile users. The fundamental challenge is how to control the spatial and temporal resolution reduction to the minimal amount that will enable mobile users to preserve the desired level of location privacy while allowing LBSs to remain effective.

The anonymity model as adopted in various fields can provide an effective approach to tackle the problem of location privacy. The aim of location anonymity is to provide location information to LBSs in a manner which would prevent them from pinpointing the exact location from which the message originated and consequently the user who created the message. The concept of k-anonymity, as proposed in [100], has been widely accepted as a feasible model for providing anonymity in relational databases. [53] first proposed

extending this approach to location privacy and [48, 82] have built on the *location k-anonymity* model proposed by [53]. Location k-anonymity emphasizes on making the user location indistinguishable from the location information of at least *k - 1* other users.

In this chapter, we present the PRIVACYGRID approach to support anonymous location-based queries in mobile information delivery systems. The design of PRIVACYGRID provides a unified and yet effective location anonymization framework for all types of location queries so that mobile users can enjoy the convenience of LBSs without revealing their exact location information. We make three unique contributions in this chapter. First, we provide a location privacy preference profile model, called location P3P, which allows mobile users to explicitly define their preferred location privacy requirements in terms of both location hiding measures (i.e., location $k$-anonymity and location $l$-diversity) and location QoS measures (i.e., maximum spatial resolution and maximum temporal resolution). Our location P3P model supports personalized and continuously changing privacy needs of a diverse user base. Second, we develop fast and effective cloaking algorithms for providing location $k$-anonymity and location $l$-diversity while maintaining the utility of LBSs. Concretely, our *dynamic expansion* technique for bottom-up grid cloaking and *dynamic reduction* technique for top-down grid cloaking provide high anonymization success rate and yet are efficient in terms of both time complexity and maintenance costs. We also propose a hybrid approach that combines the bottom-up and top-down search of location cloaking regions to further lower the average anonymization time. Last but not the least, we incorporate temporal cloaking functionality into the PRIVACYGRID location perturbation process. Deferred processing of location anonymization requests within the temporal delay constraint further enhances the anonymization success rate while maintaining the desired quality of service (QoS). Our discussion on the new capabilities required for processing anonymous location queries exhibits the benefits of using small cloaking regions for anonymous query processing. The PRIVACYGRID approach is evaluated through extensive experimentation, thus verifying that PRIVACYGRID location cloaking algorithms can

provide close to optimal location anonymity as defined by per user location P3P without introducing significant performance penalties.

## *6.2 Related Work*

Privacy has largely been the domain of policy-based solutions in the past, based on the principles of user consent, purpose binding and deployment of adequate security measures to protect user data [70]. This entails a detailed understanding of the service provider policies on the part of the user; on the other hand, service providers need to formulate privacy policies and notify users of any changes in policies leading to substantial overhead in attempting to protect user privacy. Even despite the substantial overhead, policies are merely service agreements and do not have any guarantees associated with them. Some of the prior work in location privacy has focussed exclusively on such policy-based approaches [42].

The $k$-anonymity approach to privacy protection was first developed for protecting published medical data [99]. Anonymity is a system property which implies inability to associate information with a particular individual [91]. Techniques such as generalization and suppression can be used to deanonymize or hide the values associated with some sensitive attributes (quasi-identifiers) in databases. Otherwise, the values associated with these *quasi-identifiers* may be associated with external information to uniquely identify individual records. In such a scenario, $k$-anonymity guarantees the inability of the adversary to distinguish an individual record from at least $k-1$ other records. [29, 73] provide solutions for *optimal* $k$-anonymization. Personalization of privacy requirements has attracted attention recently [48, 111]. Other related work includes anonymization of high dimensional relations [20] and extending the concept of $k$-anonymization via $l$-diversity [79], $t$-closeness [74] and $m$-invariance [112].

The concept of location $k$-anonymity was introduced in [53] where $k$ is set to be uniform for all users. The concept of personalized location $k$-anonymity with customizable QoS specifications, first introduced in [48], is adopted by several others [82, 51]. Most

popular solutions for location privacy [53, 48, 82] have adopted the trusted third party anonymization model, which has been successfully deployed in other areas such as Web browsing [3]. Two representative approaches to personalized location anonymization are the *CliqueCloak* algorithm introduced in [48] and the Casper system [82]. The *CliqueCloak* algorithm relies on the ability to locate a clique in a graph to perform location cloaking, which is expensive and shows poor performance for large $k$. The Casper approach performs the location anonymization using the quadtree-based *pyramid* data structure, allowing fast cloaking. However, due to the coarse resolution of the pyramid structure and lack of mechanisms to ensure QoS and constrain the size of the cloaking region, the cloaking areas in Casper are much larger than necessary, leading to poor QoS perceived by the users. Our experiments show that the PRIVACYGRID approach outperforms Casper and other existing location anonymization approaches in terms of efficiency and effectiveness, producing cloaking regions that meet both location privacy and location service quality requirements.

In contrast to the trusted third party anonymizer model, a couple of research projects, with Prive [51] being the most representative one, attempt to remove the trusted third party anonymizer by relying on a decentralized cooperative peer to peer model and the existence of a trusted centralized certification server. The main technical challenge handled in this work involves dynamic formation of nearby peer groups that can perform location anonymization for each other. In fact, the PRIVACYGRID approach can be easily adapted to such settings to perform the actual location cloaking among selected peer groups. Another thread of efforts is to perform location obfuscation at the mobile clients by introducing random noises or utilizing nearby landmarks [56, 41], assuming mobile clients have sufficient computation and communication resources to participate in both location anonymization and anonymous query processing tasks.

## *6.3 PRIVACYGRID: An Overview*

We assume that the LBS system powered by PRIVACYGRID consists of mobile users, a wireless network, location anonymization servers and LBS servers. Mobile users communicate with the LBS servers via one or more PRIVACYGRID location anonymization servers by establishing an authenticated and encrypted connection to the anonymization server. Each location anonymization server connects to a number of base stations, tracks the location updates of the mobile users in the range of those base stations and performs location anonymization for both location queries and location updates from these mobile users. Each location anonymization server has access to all publicly available data which can be used for ensuring location $l$-diversity for user requests.

### 6.3.1 System Architecture

The PRIVACYGRID system promotes a three-tier architecture for supporting anonymous information delivery in a mobile environment, as shown in Figure 45. The top tier is the location P3P user profile model that captures users' personalized location privacy requirements. The middle tier comprises of the location perturbation service typically provided by a trusted third party location server, specialized in location tracking and anonymization service. The third tier is dedicated to the transformation of raw location queries to anonymous location queries, enabling the processing of cloaked location queries at the individual LBS providers. Each participating LBS provider will need to provide anonymous query processing support and cooperate with the location anonymizer to provide the desired location privacy protection for consumers. In the PRIVACYGRID development, we consider the anonymous query processing component as an integral part of the solution.

Our location P3P model allows mobile users to specify when, how and with whom their location information can be shared. In addition to the standard P3P specification [7], we add four location privacy specific measures, two for location hiding constraints and two for QoS constraints. The first measure is location $k$-anonymity, which allows a mobile user

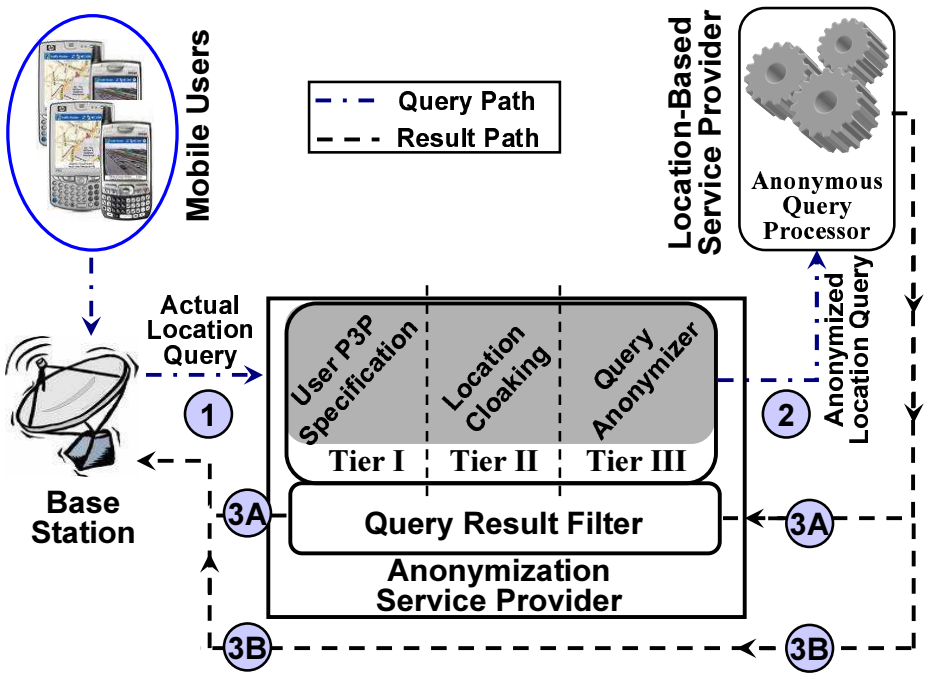



**Figure 45:** System Architecture

to control her state of being unidentifiable from a set of $k-1$ other users. The second measure is location $l$-diversity, which allows the mobile user to control her state of being unidentifiable from a set of $l$ different physical locations (such as churches, clinics, business offices). This measure can be seen as a companion measure of location $k$-anonymity and is particularly useful in reducing the risk of unwanted location inference when there are $k$ or more distinct users at a single physical location (such as a clinic or a political gathering). The third measure is *maximum spatial resolution*, which allows the mobile user to control the spatial resolution reduction within an acceptable QoS specific range. It can be changed or adjusted according to the type of location service, the time of day, month or year, and on a per message level. Similarly, the fourth measure is *maximum temporal resolution*, which controls the temporal delay acceptable for maintaining the desired QoS.

The location anonymization component anonymizes the location information from mobile users, by performing spatial and temporal cloaking, based on their P3P profiles before passing the location information to the actual LBS providers (path 2 in Figure 45). The detailed location cloaking algorithm design will be discussed in subsequent sections.

The query anonymizer component is responsible for transforming each raw location query into two components: (1) an anonymized location query by replacing the exact location of the mobile user with the location cloaking box produced by the location anonymization module; and (2) the filtering condition that will prune the candidate set of query results to produce the exact result to the original raw query posed by the mobile user. For those LBSs that offer location dependent services over public locations, such as restaurants, gas stations, offices and so forth, only anonymized location queries are passed from the location anonymizer to the respective LBS provider. Mobile users who allow their movements to be tracked by certain LBSs may use their location P3P to specify how they want their location updates to be cloaked and to which LBS servers their location updates can be provided.

Upon receiving anonymized location queries, the LBS providers will invoke the anonymous query processing module. Anonymous location query processing consists of two key steps. First, each anonymized query will be evaluated at the LBS provider to produce the set of candidate results. Second, the candidate set of query answers will need to be filtered to get exact results. There are a couple of alternative ways of performing the filtering step for anonymous query processing. For example, one can choose to have the location anonymizer as the middleman between mobile users and individual LBS providers such that the filtering task is carried out at the location anonymizer and the exact query result is generated and returned to the mobile user through the location anonymizer (path 3A in Figure 45). This approach adds additional load and bottleneck to the location anonymizer. Alternatively, filtering can be performed at the mobile client. In this case, for each location query received by the location anonymizer, the anonymized query will be passed to the LBS provider, and the filter condition will be returned to the mobile user who issued the query. The LBS provider will pass the candidate set of answers to be filtered at the client (path 3B in Figure 45). Filtering at the client will introduce additional communication and processing overhead. Thus it is critical to develop techniques that can minimize the amount of processing to be performed at the mobile client side. We will dedicate Section 6.7 for

discussing issues related to processing of anonymous location queries.

### 6.3.2 Location Privacy Requirements

In PRIVACYGRID the following requirements are considered essential for supporting anonymous location queries.

**Personalized user privacy profile:** We provide two measures for mobile users to specify their location privacy requirements: location $k$-anonymity and location $l$-diversity. The former allows a mobile user to control her state of being not identifiable from a set of $k-1$ other users. The latter allows a mobile user to control her state of being traceable to a set of at least $l$ distinct public locations, which are typically referred to as symbolic addresses. Location $l$-diversity is particularly useful in reducing the risks of unauthorized location inference when there are $k$ or more distinct users at a single physical location (such as a clinic or a church). A mobile user may change her privacy preference level ($k$ and $l$ values) as often as required or on a per message basis.

**QoS guarantees:** In order to provide effective location cloaking, PRIVACYGRID provides two QoS measures which allow a mobile user to specify critical QoS constraints. The first QoS measure is the maximum spatial resolution, indicating the amount of spatial inaccuracy the user is willing to tolerate to maintain acceptable QoS. The second QoS measure is the maximum temporal resolution, ensuring that the delay introduced for cloaking a request message should be within an acceptable time interval. By utilizing these two quality metrics, PRIVACYGRID aims at devising cloaking algorithms that will find the smallest possible cloaking region meeting desired privacy levels for each location anonymization request.

**Dynamic trade-off between privacy and quality:** In PRIVACYGRID, we stress that location perturbation algorithms should be capable of dynamically making trade-offs between privacy and QoS. Unnecessarily large cloaking boxes will lead to not only poor QoS for the mobile users but also larger result sets to be transported from the corresponding LBS

provider and higher processing costs for filtering at either the mobile client side or at the location anonymizer, inevitably leading to larger delays for obtaining useful query results.

**Efficiency and scalability:** In PRIVACYGRID, a mobile user can change her location P3P at any time. The cloaking algorithms should be effective and scalable in the presence of changing requirements on both the number of mobile users and the content of location P3P. At the same time, the cloaking algorithms must be fast and capable of keeping the perceived delays due to location anonymization to a minimum.

### 6.3.3 Basic Concepts

In this section, we define the basic concepts that are required for the subsequent discussion of the PRIVACYGRID framework.

**Universe of discourse (UoD):** We refer to the geographical area of interest as the universe of discourse (or map), which is defined by $U = Rect(x, y, w, h)$, where $x$ is the x-coordinate and $y$ is the y-coordinate of the lower left corner of a rectangular region, $w$ is the width and $h$ is the height of the universe of discourse. Basically, we consider maps which are rectangular in shape.

**Grid and grid cells:** In our framework, we map the universe of discourse $U = Rect(x, y, w, h)$ onto a grid $G$ of cells. Each grid cell is an $\alpha \times \beta$ rectangular area, where $\alpha, \beta$ are system parameters that define the cell size of the grid $G$. Formally, a grid corresponding to the universe of discourse $U$ can be defined as $G(U, \alpha, \beta) = \{A_{i,j} : 1 \leq i \leq M, 1 \leq j \leq N, A_{i,j} = Rect(x + i \cdot \alpha, y + j \cdot \beta, \alpha, \beta), M = \lceil w/\alpha \rceil, N = \lceil h/\beta \rceil\}$. $A_{i,j}$ is an $\alpha \times \beta$ rectangular area representing the grid cell that is located in the $i$th column and $j$th row of the grid $G$.

**Position to grid cell mapping:** Let $\vec{p} = (p_x, p_y)$ be the position of a moving object in the universe of discourse $U = Rect(x, y, w, h)$. Let $A_{i,j}$ denote a cell in the grid $G(U, \alpha, \beta)$. $Pmap(\vec{p})$ is a position to grid cell mapping, defined as $Pmap(\vec{p}) = A_{\lceil \frac{p_x - x}{\alpha} \rceil, \lceil \frac{p_y - y}{\beta} \rceil}$.

**Current grid cell of a moving object:** Current grid cell of a moving object is the grid cell

which contains the current position of the moving object. If $O_m$ is a moving object whose current position, denoted as $\vec{p}$, is in the Universe of Discourse $U$, then the current grid cell of the object is formally defined by $curr\_cell(O_m) = Pmap(\vec{p})$.

**User privacy preference profile:** PRIVACYGRID uses a personalized location privacy model. A user registered with the anonymization server specifies her location privacy requirements in terms of her desired user anonymity level $k$, desired location diversity level $l$, maximum spatial resolution $\{d_x, d_y\}$ and maximum temporal resolution $d_t$. Each location P3P record is of the form $\langle obj_{id}, LBS_{info}, req_{id}, k, l, \{d_x, d_y, d_t\}\rangle$, where $obj_{id}$ identifies the user, $LBS_{info}$ is optional and provides the type and the identifier of the LBS this P3P record is applied to and $req_{id}$ is used to uniquely identify a service request posed by the user with the given $obj_{id}$. We use $k = 1$ and $l = 0$ or $l = 1$ as the default setting, where neither anonymity nor diversity are considered. When $k$ and $l$ use their default settings, $d_x, d_y, d_t$ are set to unknown value *null*.

### 6.3.4 Location Anonymization Server

In PRIVACYGRID, each message $m_s$ received by the anonymizer is of the form $\langle obj_{id}, req_{id}, \{x, y, t\}, k, l, \{d_x, d_y, d_t\}\rangle$. The $obj_{id}$ and $req_{id}$ uniquely identify a message. The coordinate $(x, y)$ and the timestamp $t$ together form the three dimensional spatio-temporal location point of the mobile user who issued the message $m_s$. The parameters $\{k, l, d_x, d_y, d_t\}$ denote the location P3P specified by the mobile user who issued this request. The location anonymization server will transform the original message $m_s$ to a location perturbed message $m_t$ of the form $\langle h(obj_{id}||req_{id}), \{X : [x_s, x_e], Y : [y_s, y_e], I : [t_s, t_e]\}\rangle$, where $h$ is a secure hash function, $X : [x_s, x_e]$ and $Y : [y_s, y_e]$ denote the spatial cloaking box of the message on x-axis and y-axis respectively, such that $x_e - x, x - x_s \leq d_x$ and $y_e - y, y - y_s \leq d_y$; and $I : [t_s, t_e]$ denotes the temporal cloaking interval such that $t_e - t_s \leq d_t$. Furthermore, there are at least $k - 1$ other mobile users and at least $l$ symbolic addresses located within the same spatio-temporal cloaking box defined by

$\langle X : [x_s, x_e], Y : [y_s, y_e], I : [t_s, t_e] \rangle$. We refer to this process as spatio-temporal cloaking based message perturbation. We will describe the PRIVACYGRID spatial cloaking algorithms for finding an ideal spatial cloaking box $\langle X : [x_s, x_e], Y : [y_s, y_e] \rangle$ that meets the $k$-anonymity and $l$-diversity requirements in Section 6.4.

### 6.3.5 Evaluation Metrics

In this section, we define several metrics that will be used to evaluate the effectiveness and efficiency of PRIVACYGRID location cloaking algorithms. The anonymization success rate (ASR) and relative anonymity (relative diversity) levels are important measures for evaluating the effectiveness of the cloaking algorithms. Another useful effectiveness measure is the user location distribution (ULD) with respect to the cloaking box. It measures the strength of the location cloaking algorithm against inference attacks that attempt to guess the actual location of the mobile users with respect to the center of the cloaking region. Important efficiency measures include relative spatial resolution and message processing time.

**Anonymization Success Rate (ASR):** The primary goal of our location cloaking algorithms is to maximize the number of messages perturbed successfully while maintaining their anonymization constraints, specified by their privacy and QoS requirements. We define the anonymization success rate as the fraction of messages cloaked successfully by an algorithm with respect to the set of received anonymization requests. Let $M$ denote the set of anonymization requests issued to the system. The set of messages that are successfully perturbed can be computed by $\{m_t | m_t = f_{cloak}(m_s), m_s \in M\}$, where $f_{cloak}(m_s)$ denotes a PRIVACYGRID location cloaking algorithm. Thus, the anonymization success rate of $f_{cloak}(m_s)$ is defined as follows:

$ASR(f_{cloak}(m_s)) = \frac{|\{m_t | m_t = f_{cloak}(m_s), m_s \in M\}|}{|M|}$.

**Relative Anonymity and Relative Diversity Levels:** This metric measures the achieved anonymity and diversity levels for successfully cloaked messages normalized by the anonymity

level $k$ and diversity level $l$ in the mobile user's location P3P. Intuitively, Relative Anonymity Level (RAL) measures the ratio of anonymity achieved by the cloaking algorithm to the user specified $k$-anonymity level, i.e., $\frac{k'}{k}$ and Relative Diversity Level (RDL) provides a similar measure for $l$-diversity $\frac{l'}{l}$, where $k, l$ denote the user-defined values for a message $m_s$ and $k', l'$ denote the actual values obtained for the perturbed message $m_t(k' \geq k, l' \geq l)$. Note that for successful anonymization relative anonymity level cannot go below 1. Although, the location cloaking algorithms aim at obtaining higher anonymity for the same cloaking area, excessive anonymity achieved at the cost of cloaking the location to a much larger region leads to poor QoS and costly processing of anonymous queries. Hence, the lower the relative anonymity and relative diversity levels, the more effective the cloaking algorithm.

**Relative Spatial Resolution (RSR):** This metric measures the ability of a cloaking algorithm to provide the smallest cloaking area that meets the $k$-anonymity and $l$-diversity requirements. Given a message $m_s$ and its perturbed version $m_t$, we can measure the RSR by using the minimum spatial cloaking area as calculated by the cloaking algorithm. We define the RSR of a cloaking algorithm over a set of perturbed messages $T$ as follows: $RSR = \frac{1}{|T|} \sum_{m_t = f_{cloak}(m_s) \in T} \sqrt{\frac{2 \cdot m_s.d_x \cdot 2 \cdot m_s.d_y}{||B_{cl}(m_t).X|| \cdot ||B_{cl}(m_t).Y||}}$, where $(d_x, d_y)$ denote the maximal spatial resolution constraints for message $m_s$ and $(B_{cl}(m_t).X, B_{cl}(m_t).Y)$ represent the dimensions of the cloaking box $B_{cl}(m_t)$. Higher relative spatial resolution measure implies that the cloaked spatial region is smaller relative to the user-specified maximum spatial resolution area and the cloaking algorithm is more effective.

**User Location Distribution(ULD):** This metric is used to measure the level of difficulty in inferencing the location of a user within the cloaking region shared by $k$ users. We determine the user location distribution within the cloaked area by measuring the normalized distance of the actual user position to the center of the cloaked area for each successfully anonymized message. A uniform user location distribution implies the algorithm is more effective in terms of robustness against aforementioned inference attacks as the actual user

location may lie anywhere within the cloaked region.

**Message Anonymization Time:** This metric measures the run-time performance of the cloaking algorithm in terms of time complexity. Efficient cloaking implies that the cloaking algorithm spends less processing time to perturb messages.

## *6.4 PRIVACYGRID Spatial Cloaking Algorithms*

In this section we introduce basic data structures and two dynamic grid-based location cloaking algorithms: bottom-up spatial cloaking and top-down spatial cloaking, both aiming at finding the smallest spatial cloaking box given the location of a mobile user. Ideally, this means that there exists no smaller spatial cloaking box that satisfies both location $k$-anonymity and location $l$-diversity requirements as well as the maximum spatio-temporal resolution constraints defined in the users' location P3P.

### 6.4.1 Data Structures

In PRIVACYGRID the entire UoD is divided into grid cells of $\alpha \times \beta$ size by superimposing a grid on top of the UoD. $\alpha$ and $\beta$ are system-controlled parameters that can be tuned based on a number of factors, such as the cloaking speed, granularity of cloaking boxes and average size of user-defined maximum spatial resolutions. Figure 46 illustrates the PRIVACYGRID *Index (PI)* and the *Cell Object Count Map (COCM)* data structures.

**PRIVACYGRID Index (PI):** The $PI$ data structure allows for fast and efficient computation of object counts belonging to a particular region of the UoD. Figure 46 illustrates the composition and construction of this grid-based object index. The mobile object index and the still object index share the same data structure though maintained separately.

**Cell Object Count Map (COCM):** In addition to the mapping of each object to its current grid cell maintained by $PI$, we use this data structure to keep a count of the number of mobile objects and a count of the number of static objects (symbolic addresses, such as gas stations, restaurants, offices, and so forth) located in each grid cell. Maintaining this data structure allows for fast computation of the total number of mobile users and the total

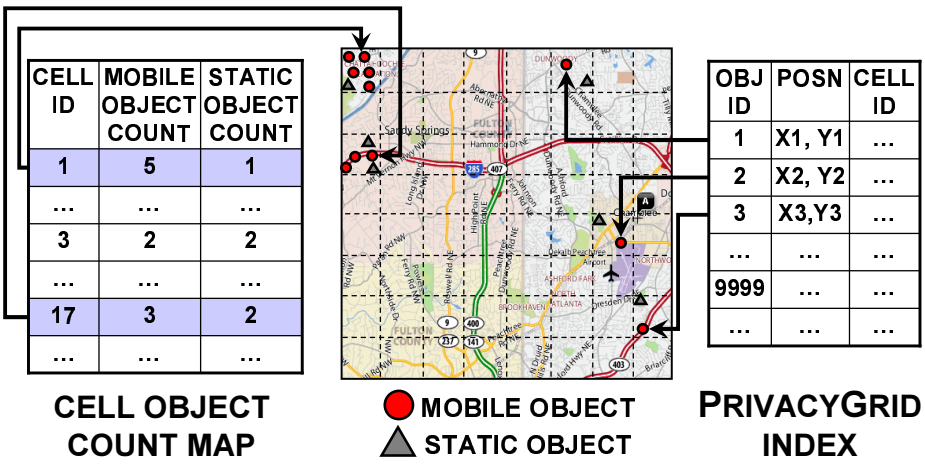



**Figure 46:** Data Structures for PRIVACYGRID

number of static objects located in a given spatial area. For each grid cell, the count of static objects remains unchanged most of the time. However, the count of mobile objects may change as mobile users move from one cell to another.

### 6.4.2 Bottom-Up Grid Cloaking

The bottom-up grid cloaking approach starts the cloaking process by taking the base cell containing the mobile object from which the cloaking request has originated as the candidate cloaking area. It performs two checks for each message with $k$ or $l$ higher than one in order to determine whether this candidate cloaking area meets the privacy and QoS requirements to be qualified as the ideal cloaking region. The first check is to determine if the current cell meets the user-specified maximum spatial resolution constraints. A second check looks up the cell object count map to determine if $k$-anonymity and $l$-diversity requirements are met. If the second check is successful, the candidate cloaking area will be chosen as the cloaking region. If not, the algorithm will start the cell expansion process to enlarge the candidate cloaking area to neighboring cells. The cell expansion process stops when both $k$-anonymity and $l$-diversity requirements for the cloaked message are met.

The core idea behind bottom-up cloaking is the execution of dynamic cell expansion when the candidate cloaking region fails to meet the location privacy and QoS constraints.

Dynamic cell expansion takes an opportunistic approach to expand the candidate cloaking region to any of the four neighboring set of cells.

The decision on which of the four cells to choose first is based on the object counts; the neighboring cell(s) with the highest object count will be chosen for expansion, generating the new candidate cloaking box. Each candidate cloaking box is composed of a set of adjacent cells and is encoded by the row and column index of the these selected cells. We maintain the *selected rows* and the *selected columns* for all candidate cloaking boxes in order to infer the selected cells of the final cloaking area. The current candidate cloaking box may be expanded in any direction (*North, South, East or West*) by adding the row above the uppermost selected row (or below the lowermost selected row) or the column to the right of the rightmost selected column (or to the left of the leftmost selected column), thus dynamically building the cell-based cloaking box by selecting and adding the rows or the columns which lead to the maximum object count collectively.

For every odd iteration, the algorithm determines whether to add a row or a column as the cloaking area may be expanded in any direction. For even iterations, the algorithm expands the cloaking area, depending on whether a row or column was added in the previous iteration, in order to ensure that no vertical or horizontal skew is introduced. For example, if the algorithm added a row during the previous iteration, the current iteration would expand the cloaking box by addition of an adjacent column. The cell expansion steps are recursively repeated as long as the sum of object counts in *all* cells in selected rows and columns is less than the required $k$-anonymity and $l$-diversity levels. Upon meeting both the privacy and the QoS requirements, the algorithm uses the selected rows and columns to determine the grid cells forming the final cloaking area.

Figure 47 presents a walkthrough of the bottom-up dynamic expansion by example. In this example, we assume the user-defined $k$ value is 20 and $l$ value is 3. The cloaking request originates from the shaded cell with the mobile object count (number at top-left in each cell in Figure 47) of 6 and the still object count (number at bottom-right in each

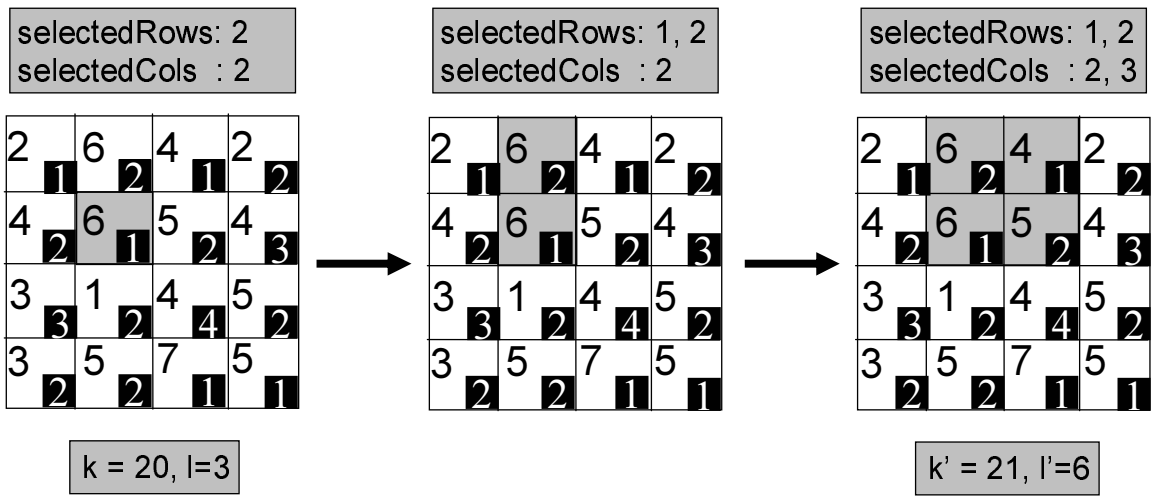



**Figure 47:** Bottom-Up Dynamic Expansion Example

cell in Figure 47) of 1. It is located at the second row and the second column in the grid. We encode this candidate cloaking region by assigning the value of 2 to both *selectedRows* and *selectedCols* respectively. Clearly, this cell fails to meet both the $k$-anonymity and the $l$-diversity requirements. The algorithm starts the dynamic cell expansion process from the current cell. All neighboring cells of the shaded cell are considered. Given that the first row to the north increments the mobile object count to 12, the highest among all four neighboring cells, it is chosen to be the first cell for expansion. We add the row number 1 into the *selectedRows* to encode the new candidate cloaking region. Even though the total still object count in this candidate cloaking box is 3, satisfying the $l$-diversity requirement, the total mobile object count of 12 does not meet the user-specified $k$-anonymity requirement of 20. Thus the algorithm starts the next iteration of cell expansion. In this iteration, we choose one of the two neighboring columns of the candidate area to expand. We first consider the column to the left (first column in grid), which is not sufficient to meet the privacy requirements. Then we consider the column to the right (the third column in grid) which provides a cloaking area with the object count of k'=21, sufficient to meet the $k$-anonymity requirement. Thus the algorithm terminates and returns $selectedRows = \{1, 2\}$ and $selectedCols = \{2, 3\}$.

In fact, there are two ways in which cell expansion can be performed: (1) static cell expansion based on a pre-defined pattern, such as the quadtree-based grid expansion [82] or (2) dynamic cell expansion that opportunistically determines appropriate neighboring cells to expand the candidate cloaking region at run time. It is interesting to note that the static expansion approach promotes static cloaking following a pre-defined structure. For example, the pyramid approach in [82] uses the quadtree-based pyramid structure and some steps may expand the cloaking area to a pre-defined parent cell along the pyramid hierarchy, quadrupling the cloaked area, limiting the ability of the algorithm to explore all options of varying granularity. Though such a static cloaking approach is simple and fast, it suffers from a number of weaknesses. For example, pyramid cloaking expands the cloaking area to two or four times of the current size at each iteration, leading to a much bigger cloaking area and a much higher anonymity level than required, which hurts the QoS provided to the user and results in low anonymization success rate. In contrast to the static cloaking approach that selects the cloaking area using a pre-built cell composition structure, the dynamic expansion approach opportunistically expands the cloaking area, enabling the algorithm to quickly locate an ideal cloaking area that meets privacy and QoS requirements. Figure 48 shows pyramid expansion cloaking for the same example used in Figure 47. Clearly, pyramid expansion results in a much larger cloaking area with a much higher anonymity level than required. In contrast, the cloaking area produced by the dynamic bottom-up approach is much smaller as shown in Figure 47.

### 6.4.3 Top-Down Grid Cloaking

In some scenarios, a top-down cloaking approach performs anonymization faster compared to the bottom-up approach. For example, high $k$-anonymity and low maximal spatial resolution constraints may help the system quickly locate appropriate cloaking areas by using a top-down dynamic reduction approach as explained below. In PRIVACYGRID, we

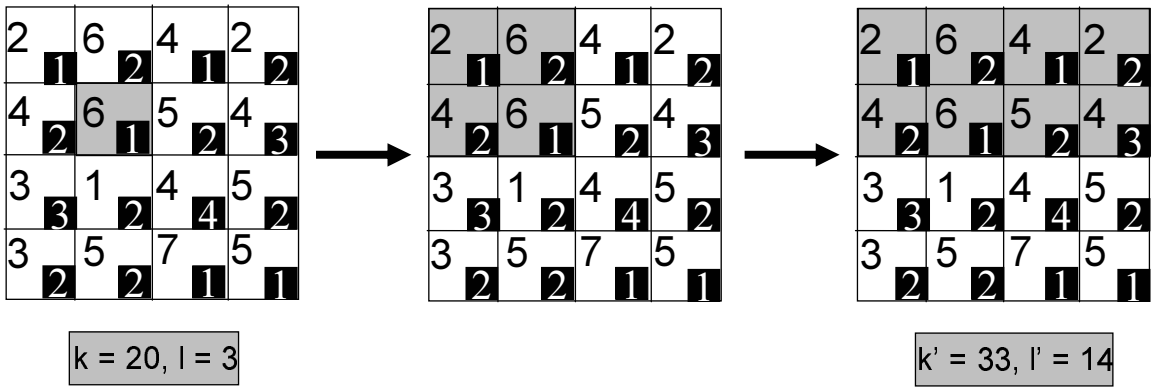



**Figure 48:** Pyramid Expansion Example

design the top-down dynamic grid cloaking algorithm by utilizing the user-specified maximum spatial resolution. We first find the largest grid cell region within the user-specified maximum spatial resolution area, and encode the candidate cloaking area by a set of *selectedRows* and *selectedCols* in a similar manner as is done in the bottom-up approach. If the largest possible candidate cloaking box fails to meet the desired privacy requirements, the message cannot be cloaked using user-defined privacy and QoS requirements and the algorithm terminates. Otherwise, the top-down cloaking approach starts searching for the smallest possible cloaking box that meets the $k$-anonymity and $l$-diversity requirements by iteratively removing either an outermost row or column with the lowest object count from the candidate cloaking area. This iterative process shrinks the candidate cloaking box along one of the four directions and terminates when object counts in candidate cloaking area fall below the privacy requirement.

Figure 49 displays an example walkthrough of the top-down dynamic cloaking algorithm. Recall the previous example where a mobile user in the cell at the intersection of the second row and second column issued a location anonymization request. The shaded area in the leftmost figure displays the largest possible cloaking area computed based on the user-specified maximum spatial resolution. Given that the mobile object count is 35 and the still object count is 18, cell reduction is performed repeatedly by first removing the third row (lowest mobile object count) and then removing the first column. The final

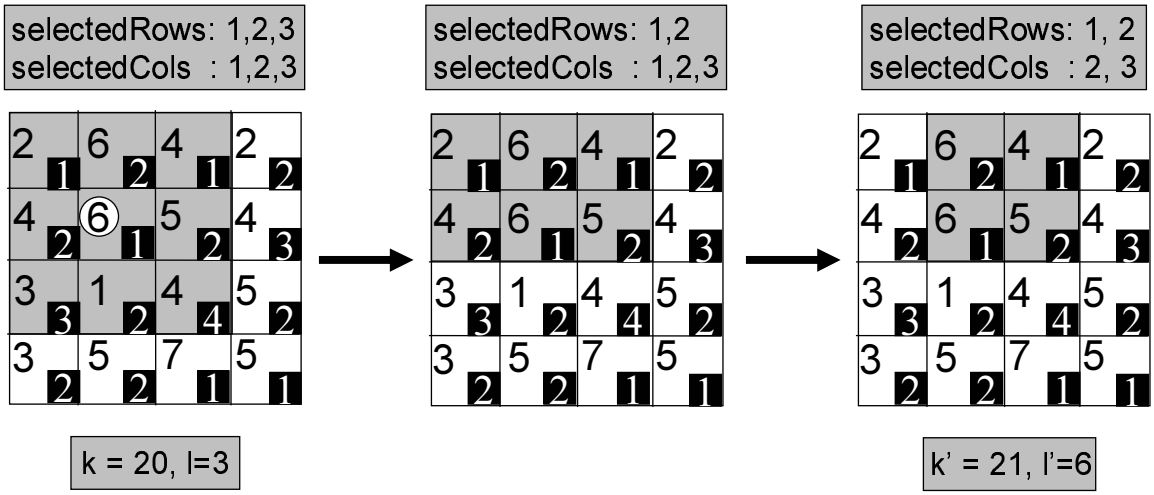



**Figure 49:** Top-Down Dynamic Reduction Example

cloaking box consists of the four cells marked by the first two rows and the second and third columns, with $k'$=21 and $l'$=6.

## *6.5 Analysis*

We provide a brief analysis of the evaluation metrics described in Section 3.6.2 for the cloaking algorithms described in the previous section. The analysis for the bottom-up and top-down dynamic approaches is similar and a comparison is made with the pyramid approach [82]. We deal with a UoD having $n$ mobile users and a grid of $a$ rows $\times$ $b$ columns superimposed on top of the UoD with cell size $\alpha \times \beta$. Assuming uniform distribution of objects over the UoD, object density per cell can be calculated as $\gamma = \frac{n}{a \times b}$. Assume the location anonymization requests are made with desired anonymity levels $i \epsilon$ [2, $k$], where $2^{m-1} \leq k < 2^m$. The achieved anonymity level depends on the number of selected cells and can be calculated as $x \times \gamma$, where $x$ is the number of cells selected by the cloaking algorithm to satisfy the $k$-anonymity requirement.

Assuming cloaking requests are uniformly distributed over the range of $k$ values,

$$RAL = \frac{1}{k-1} \times [\frac{x_1 \times \gamma}{2} + \frac{x_2 \times \gamma}{3} + ... + \frac{x_{k-1} \times \gamma}{k}], \tag{26}$$

where $\forall\ i\ \epsilon$ [2,$k$], $x_{i-1}\ \epsilon\ 2^s$, $s \geq 0$ for the pyramid approach as the number of cells selected by the approach is a power of 2. For our dynamic cloaking approaches, $\forall\ i\ \epsilon$ [2,$k$], $x_{i-1}\ \epsilon\ s^2$ or $s(s+1)$, $s \geq 1$ assuming that no skew is introduced in any direction, $\alpha \simeq \beta$ and $d_x \simeq d_y$ to allow number of selected rows to be equal to number of selected columns (leading to $s^2$ cells) or differ by one (leading to $s(s+1)$ cells).

$$RAL = \frac{1}{k-1}\sum_{i=2}^{k}[\frac{x_{i-1} \times \gamma}{i}], \tag{27}$$

where $\forall\ i\ \epsilon$ [2,$k$], $x_{i-1} \times \gamma \geq i$ in order to satisfy the anonymity requirement. Similarly assuming same maximum spatial resolution values $(d_x, d_y)$ for all requests,

$$RSR = \frac{1}{k-1}\sum_{i=2}^{k}\sqrt{[\frac{4 \times d_x \times d_y}{x_{i-1} \times \alpha \times \beta}]} \tag{28}$$

### *6.6 Hybrid Cloaking*

An obvious enhancement to bottom-up and top-down cloaking algorithms is the hybrid approach that takes advantage of the strengths of both approaches to produce a cloaking algorithm that *runs faster* than either of them. There are several ways to combine the bottom-up and top-down methods. In the first prototype of PRIVACYGRID we adopt a most straightforward approach. The main idea is to provide guidelines on how to appropriately decide whether to proceed in a bottom-up or a top-down manner upon receiving a message cloaking request. For lower $k$-anonymity level and higher maximum spatial resolution values, the algorithm will benefit by proceeding in a bottom-up manner. On the other hand, for higher $k$-anonymity level and lower maximum spatial resolution values, the top-down approach clearly works faster than the bottom-up approach for finding the ideal cloaking box. Hence, the ability of the hybrid approach to identify whether it should proceed in a bottom-up or top-down manner upon receiving a cloaking request is crucial to its effectiveness. We provide some guidelines through a formal analysis of the hybrid cloaking algorithm in [24].

Consider the UoD with a grid comprising of cells of size $\alpha \times \beta$ superimposed on top of it. We assume that objects are uniformly distributed over the UoD and each cell has $\gamma$ objects on average. The bottom-up (or top-down) approach advocates addition (or removal) of rows and columns alternately. We assume that the final cloaking area consists of an equal number of rows and columns. Given an anonymization request with anonymity level $k$, we conclude that $c$=$\sqrt{\frac{k}{\gamma}}$, where $c^2$ is the average number of cells estimated to meet our anonymity requirements. Consequently, given a base cell, we need to have $c$ rows and $c$ columns to form the required cloaking area. Assume that the largest cloaking area, as defined by the maximum spatial resolution values $(d_x, d_y)$, consists of $max_r$ rows and $max_c$ columns which can be quantified as below.

$$max_r = 2 \times \left\lfloor \frac{d_y}{\beta} \right\rfloor + 1; max_c = 2 \times \left\lfloor \frac{d_x}{\alpha} \right\rfloor + 1 \tag{29}$$

The bottom-up approach starts with a singe cell; locating an ideal cloaking region is expected to add only $(c-1)$ rows and $(c-1)$ columns on average, where $c$=$\sqrt{\frac{k}{\gamma}}$. The top-down approach starts with $max_r$ rows and $max_c$ columns and is expected to remove $(max_r - c)$ rows and $(max_c - c)$ columns on average. Hence the expected number of iterations performed by the bottom-up approach ($i_{bu}$) and top-down approach ($i_{td}$) is,

$$i_{bu} = 2 \times \left\{ \sqrt{\frac{k}{\gamma}} - 1 \right\}; i_{td} = \left\{ max_r - \sqrt{\frac{k}{\gamma}} \right\} + \left\{ max_c - \sqrt{\frac{k}{\gamma}} \right\} \tag{30}$$

The initial iterations performed by the top-down approach, on average, require more computation when compared with the initial iterations performed by the bottom-up approach as the top-down iterations are initially acting on a much larger number of rows and columns compared to the bottom-up approach. Let the average cost of iterations performed by the top-down approach be $\delta$ times the average cost of iterations performed by the bottom-up approach. From the above equations we can determine the cost of proceeding in

a bottom-up manner as,

$$T_{bu} = 2 \cdot \sqrt{\frac{k}{\gamma}} \cdot Cost_{bu} \tag{31}$$

where $Cost_{bu}$ is the average cost of a single bottom-up iteration. Similarly, if the average cost of one top-down iteration is $Cost_{td} = \delta \times Cost_{bu}$, then

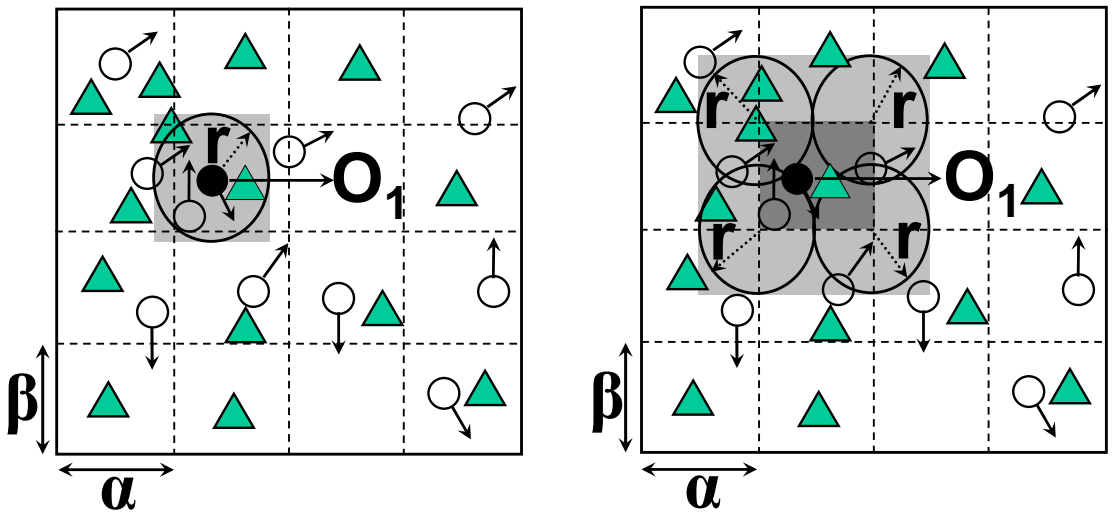
$$T_{td} = \left\{ max_r - \sqrt{\frac{k}{\gamma}} + max_c - \sqrt{\frac{k}{\gamma}} \right\} \cdot \delta \cdot Cost_{bu} \tag{32}$$

For any anonymization request, the hybrid cloaking algorithm proceeds in a top-down manner if $T_{td} < T_{bu}$ or equivalently if $\delta < \frac{2 \cdot \sqrt{\frac{k}{\gamma}}}{max_r + max_c - 2 \cdot \sqrt{\frac{k}{\gamma}}}$, otherwise, it proceeds in a bottom-up manner.

## *6.7 Processing Anonymous Queries*

In PRIVACYGRID, each location query will be sanitized through the location anonymization server before proceeding to the relevant LBS provider. The location anonymization engine will transform a raw location query into two components: anonymized query and privacy sensitive filter. The anonymized query can be submitted to the LBS providers by either the mobile user who issued the original query or by the anonymizer. However, the privacy-sensitive filter will be kept either at the location anonymization engine or on the mobile client side. Upon receiving an anonymous query, the LBS provider will invoke the anonymous location query processing engine residing at the LBS provider. Based on processing logic, we divide anonymous location queries into two classes: location anonymous queries over static objects (public location data) and location anonymous queries over moving objects (privacy sensitive location data). In either case, instead of exact answers, the anonymous location query processor will return approximate query answers that include the exact answer. The exact answer will be computed over the minimal approximate answer either at the mobile client or at the location anonymizer, which then forwards the exact answer to the mobile client.

Anonymous location query processing poses two unique challenges. First, we must




(a) Actual Query (b) Anonymized Query

**Figure 50:** Anonymous Query Processing

produce the minimal set of approximate answers, aiming at minimizing the amount of additional communication and computation cost due to the location privacy support. Second, with anonymous location queries, the exact query result must be delivered to the mobile user. This may be done through the trusted location anonymizer, which performs the post-processing and forwards the exact answer to the mobile user. Alternatively, the minimal approximate answer can be forwarded directly to the mobile client by the LBS provider through the subset of base stations that cover the region of the minimal query answers. The base stations may broadcast the set of approximate answers with the secure query identifier $-$ the hash of $user_{id}$ and $req_{id}$ (recall Section 6.3.4); only the mobile user who knows the *secret* identifier will be able to read the result set and perform the post-processing to produce the exact answer.

Most of the spatio-temporal query processing techniques developed in the mobile data management field to date cannot be applied directly to anonymous query processing. We briefly describe below the anonymous query processing mechanisms required at the LBS server. In order to process range queries associated with cloaked spatial regions instead of spatial points and produce the minimal set of approximate answers for each anonymous

query, the anonymous query processor needs to process each query in three steps: (1) determining the anonymous query minimum bounding rectangle (anonymous MBR) that contains the minimal set of approximate answers, (2) transforming the anonymous location query with anonymous MBR to an anonymity-aware range query, and (3) executing the range query using the traditional spatial query processor to produce the minimal set of approximate answers. Figure 50 illustrates this process using an example, where a moving object $O_1$ issues a query $Q$, requesting for some static objects (e.g. restaurants) within the distance $r$ from its current position. Figure 50(a) shows the Minimum Bounding Rectangle (light grey rectangle) which forms the exact result set of the query $Q$ using a traditional mobile query processor. Figure 50(b) shows the anonymous MBR (light grey rectangle) which produces the minimal set of approximate answers for the perturbed version of $Q$. The cloaked query region produced by the location anonymization server is shown as a dark grey rectangle contained inside the anonymous MBR. The mobile object $O_1$ could be present anywhere within the cloaked query region. Thus the computation of anonymous MBR will need to consider the four corner points of the cloaked rectangle and the entire region within a maximum distance $r$ from each corner point to ensure that the anonymous MBR includes the exact answer of the query, as long as $O_1$ lies within the cloaked region. Different formulae will be required for computing anonymous MBR for different types of location queries. For example, [82] presents an efficient algorithm for privacy-aware processing of nearest neighbor queries (kNN).

## *6.8 Experimental Evaluation*

We divide the experimental evaluation of PRIVACYGRID into two components: the effectiveness of our cloaking algorithms in terms of privacy and quality requirements and their performance in terms of time complexity and scalability. Before reporting our experimental results, we first describe the experimental setup, including the road-network based mobile object simulator used in the experiments.

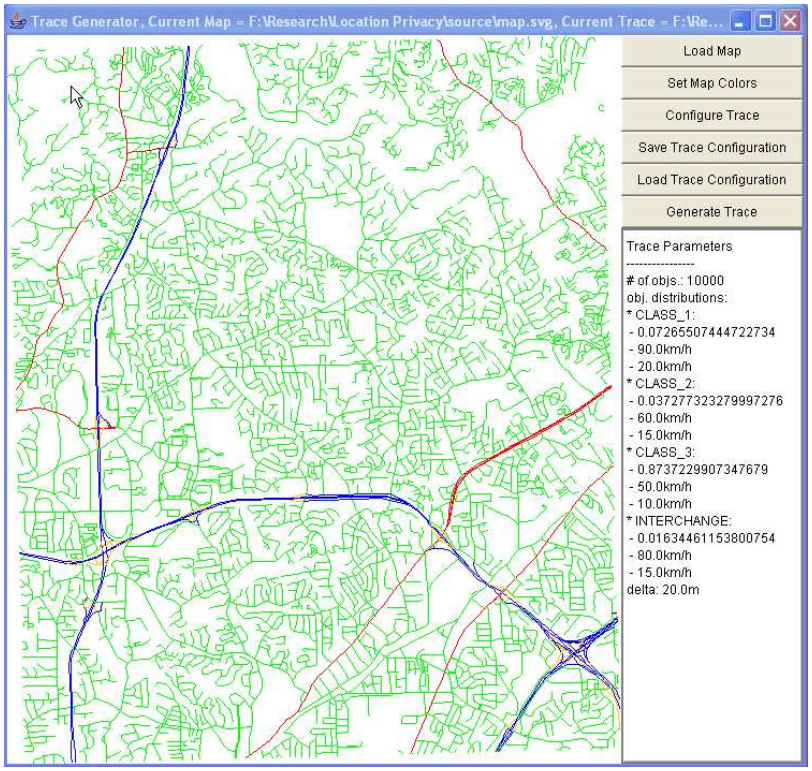



**Figure 51:** Simulator for Experimental Setup

### 6.8.1 Experimental Setup

We extend the simulator from [48] to evaluate the effectiveness and performance of PRIVACYGRID cloaking algorithms. The simulator generates a trace of cars moving on a real-world road network, obtained from maps available at the National Mapping Division of the USGS [8], and generates requests based on the position information from the trace. The simulator extracts the road network based on three types of roads – *expressway*, *arterial* and *collector* roads. Our experimentation uses a map from the Chamblee region of Georgia (Figure 51), which covers an area of approximately 168 $km^2$, to generate traces for a one hour duration. Traffic volume data from [53] is used, generating a set of 10,000 cars on the road network for Chamblee. Table 4 lists mean speeds, standard deviation and traffic volume values for each road type. Cars are randomly placed on the road network according to the traffic densities, start moving on the roads and proceed in a random direction at the intersections. The simulator attempts to keep the number of cars on each type of road

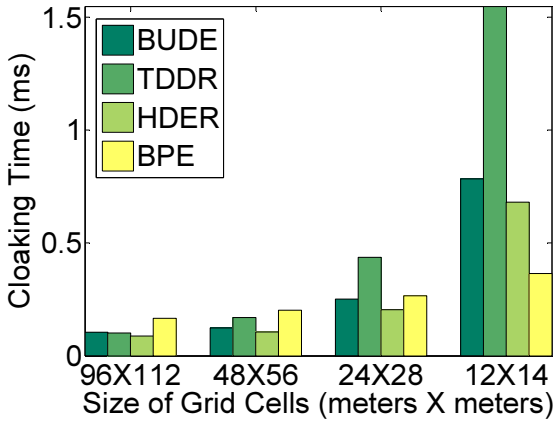
**Table 4:** Evaluation Setup Parameters

| **Road type** | **Expressway** | **Arterial** | **Collector** |
|---|---|---|---|
| **Mean speed(km/h)** | 90 | 60 | 50 |
| **Std. dev.(km/h)** | 20 | 15 | 10 |
| **Traffic data (cars/h)** | 2916.6 | 916.6 | 250 |

constant with time. Each car generates a set of messages during the simulation. By default, each message specifies an anonymity level $k$ from the range $[2, 150]$ using a Zipf parameter of 0.6 favoring higher $k$ values. Our experiments do not consider $l$-diversity requirements, but can be easily extended to measure relevant values. The maximum spatial resolution (or spatial tolerance) values of the messages are selected independently using normal distribution with default mean spatial resolution of $600m$ and $5\%$ standard deviation. Though all parameters take their default values if not stated otherwise, the settings of many parameters are changed in different experiments to show the impact of these parameters on the effectiveness and efficiency of the algorithms.

### 6.8.2 Experimental Results

Our experimental evaluation of the PRIVACYGRID algorithms consists of two parts. First, we evaluate the effectiveness of the location anonymization algorithms by measuring success rate, relative anonymity level, relative spatial resolution, average cloaking time, user location distribution and observe how these parameters behave when we vary the settings of a number of parameters, such as grid cell size, anonymity level $k$ and maximum spatial resolution $\{d_x, d_y\}$. Then we evaluate the scalability of the algorithms in terms of cloaking time and update cost by varying the number of mobile users. We briefly describe the impact of incorporating temporal cloaking on the anonymization success rate of our dynamic approaches. Our results show that the PRIVACYGRID dynamic grid cloaking algorithms are fast, effective, scalable and outperform other location cloaking approaches in terms of both anonymization success rate and cloaking QoS in the presence of a larger range of $k$ values.




(a) Anonymization Time

| | Cell Size (meters) | Success Rate | RAL | RSR |
|---|---|---|---|---|
| BUDE | 24 X 28 | 93.9% | 1.0001 | 3.276 |
| | 48X 56 | 92.8% | 1.0009 | 3.291 |
| TDDR | 24 X 28 | 93.9% | 1.001 | 3.296 |
| | 48X 56 | 92.9% | 1.004 | 3.243 |
| HDER | 24 X 28 | 93.9% | 1.0001 | 3.305 |
| | 48X 56 | 92.9% | 1.001 | 3.306 |
| BPE | 24 X 28 | 40.9% | 1.17 | 2.366 |
| | 48X 56 | 40.9% | 1.176 | 2.327 |

(b) Other Metrics

**Figure 52:** Results with Varying Size of Grid Cells

#### *6.8.2.1 Varying Size of Grid Cells*

This set of experiments aims at measuring cloaking performance obtained by using different settings of grid cell size. Figure 52 shows the results measured for different settings of grid cell size which are equivalent to different settings for the grid size, ranging from $128 \times 128$ cells to $1024 \times 1024$ cells. The user-defined anonymity levels for this set of experiments are chosen in the range [10 – 50] with a Zipf distribution using parameter 0.6 favoring higher $k$ values.

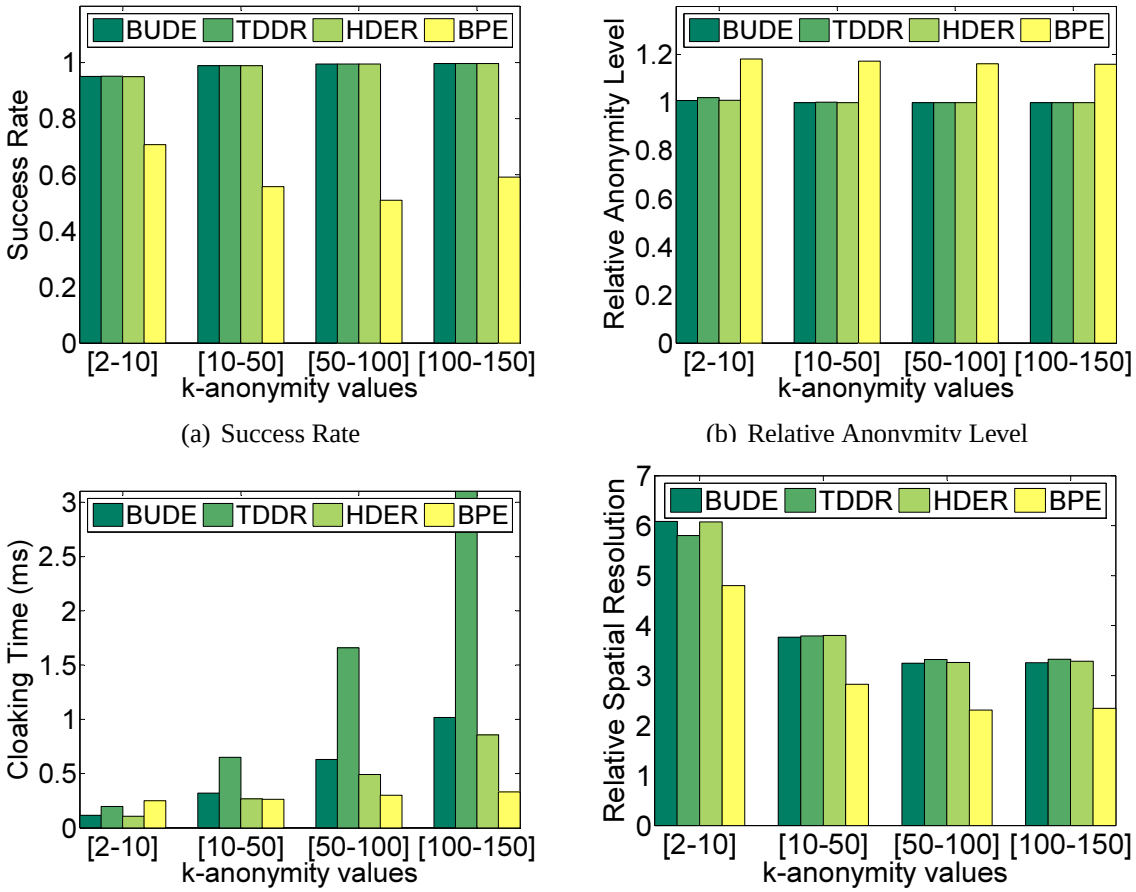
Figure 52(a) shows that the basic pyramid expansion (BPE) is fast in terms of cloaking time and the cloaking time does not increase significantly with the decrease in the size of grid cells. We implement the basic location anonymizer using the pyramid approach as described in [82]. Due to the fine granularity of the grid structure and consequently small cell sizes, the adaptive location anonymizer [82] does not work well due to frequent splitting and merging of cells and experiences inferior performance compared to the basic anonymizer. Except for the smallest grid size, both bottom-up dynamic expansion (BUDE) and top-down dynamic reduction (TDDR) almost match the performance of BPE. More rows (or columns) need to be added (or removed) to obtain ideal cloaking regions but maintenance of data structures is less expensive for these approaches. Interesting to note is that the actual cloaking time of all dynamic approaches is still below 1.5 ms in all cases

and such low delays are hardly perceivable. Hybrid dynamic expansion reduction (HDER) performs better than both bottom-up and top-down approach, adapting appropriately to each message, by deciding whether to proceed in a bottom-up or top-down fashion.

From Figure 52(b) we observe two interesting results. First, the success rate, the relative anonymity level (RAL) and the relative spatial resolution (RSR) do not change much as we vary the size of grid cells. Second, given a fixed grid cell size, say $[24m \times 28m]$, we see sharp differences when comparing BPE with the three dynamic grid cloaking approaches. BPE, though marginally faster for the smallest grid cell sizes (recall Figure 52(a)), has only 41% of the messages being anonymized successfully when QoS measures are considered, while all the dynamic approaches have similar but much higher rate of success ($> 92.8\%$). All the dynamic approaches give low relative anonymity levels, which are close to one, whereas the BPE approach has about 17% higher relative anonymity level, indicating that it might be cloaking requests to unnecessarily larger spatial regions. This is confirmed by the relative spatial resolution (RSR) measurement, which is about 40% higher for the dynamic approaches when compared to BPE. We use grid cells of size $[24m \times 28m]$ for further evaluation.

#### *6.8.2.2 Varying User-defined Anonymity Level $k$*

This set of experiments evaluates cloaking performance with varying anonymity level $k$ for various ranges: [2–10], [10–50], [50–100] and [100–150]. Maximum spatial resolution values for the anonymity ranges are 400 m, 800 m, 1200 m and 1600 m (mean values with 5% standard deviation) respectively and are chosen to be large enough to theoretically allow cloaking of a large fraction of the messages. Figure 53 shows that BPE is able to cloak only around 70% of the messages with anonymity level $k$ set in the range of [2–10] and the success rate falls further to 50-60% with increasing $k$ values. In contrast, the dynamic approaches cloak 95-99.6% of the messages within user-defined maximum spatial resolution values (Figure 53(a)). From Figure 53(b), we see that BPE results in higher relative




(a) Success Rate

(b) Relative Anonymity Level

(c) Anonymization Time

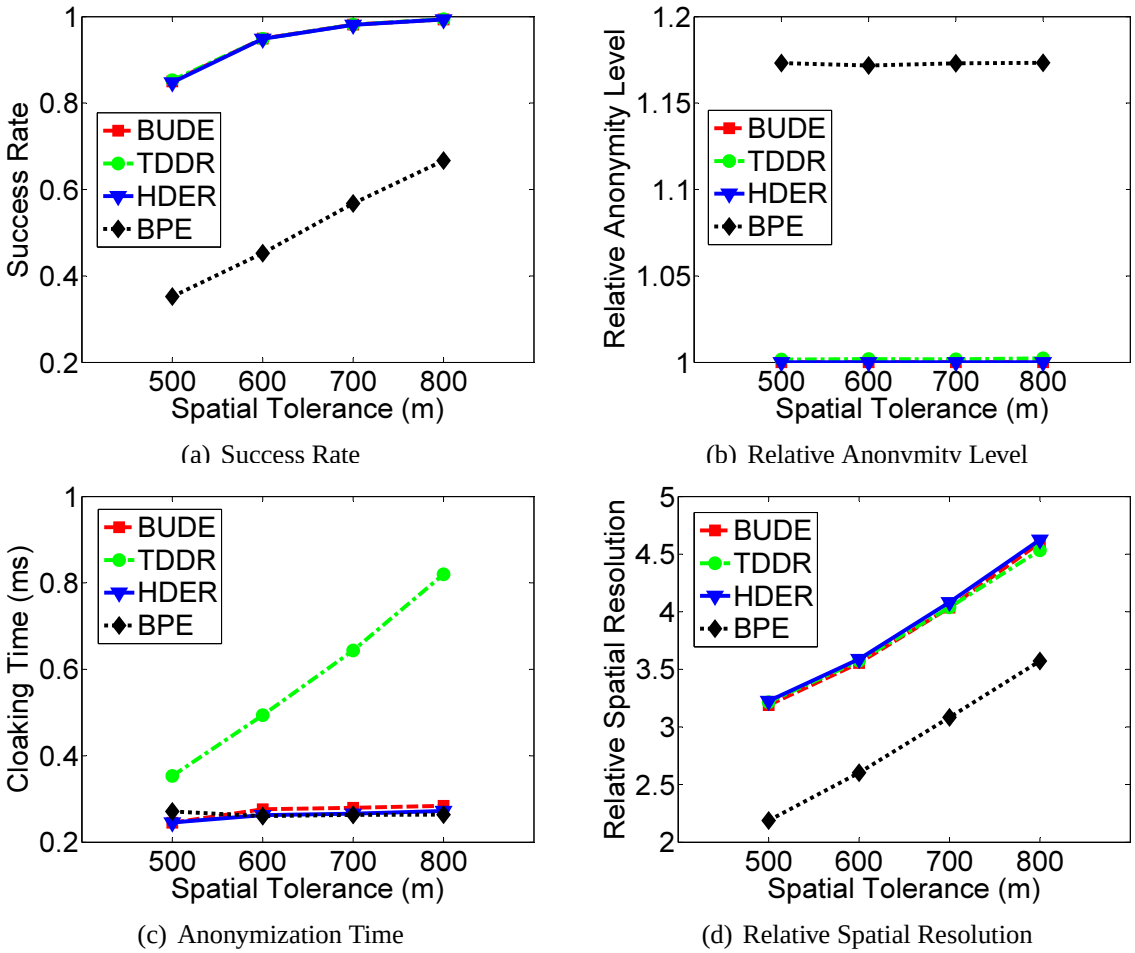
(d) Relative Spatial Resolution

**Figure 53:** Results with Varying Anonymity Levels

anonymity level but all dynamic cloaking approaches have relative anonymity levels close to one, indicating that the anonymity levels obtained for all perturbed messages are very close to the user-defined $k$.

Figure 53(c) shows the impact of varying $k$ on the cloaking time of all algorithms. BPE is the fastest and its cloaking time does not increase much with the increase in the user-defined $k$ values. Though all dynamic cloaking algorithms will incur relatively higher cloaking time with increasing $k$ values, the amount of increase in cloaking time for BUDE and HDER is lower when compared to TDDR. It is important to note that the cloaking time for the worst case (where the top down approach is used) is still below 3.5 ms for $k$ values in [100–150].

Figure 53(d) displays the impact of changing $k$ values on relative spatial resolution




(a) Success Rate (b) Relative Anonymity Level

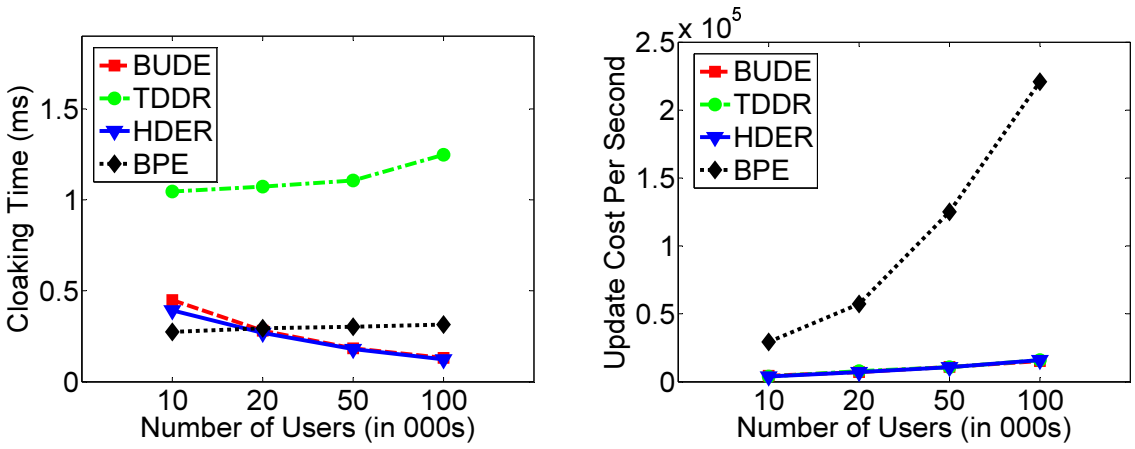
(c) Anonymization Time (d) Relative Spatial Resolution

**Figure 54:** Results with Varying Maximum Spatial Resolution

(RSR) obtained for the perturbed messages. Clearly, the dynamic cloaking algorithms have considerably higher RSR (25-35%) than BPE approach for all $k$ values, though RSR values decrease as the $k$ values become larger.

#### *6.8.2.3 Varying Maximum Spatial Resolution Values*

This set of experiments examines the performance of the algorithms by varying the maximum spatial resolution settings; messages are generated with anonymity level $k$ from the range [10–50] with Zipf distribution using parameter 0.6, favoring messages with higher $k$ values. We vary the maximum spatial resolution value from 500 m to 800 m (mean values with 5% standard deviation) and examine the effect of different settings of maximum spatial resolution on the effectiveness of the different approaches. Figure 54 displays




(a) Anonymization Time

(b) Update Costs

**Figure 55:** Results with Varying Number of Users

the results. The dynamic approaches are able to cloak all messages which can be theoretically cloaked for each maximum spatial resolution value, whereas BPE fails to cloak a large number of messages (50-60% less as shown in Figure 54(a)). Figure 54(b) shows that the relative anonymity levels for all cloaking algorithms do not change much when the user-defined maximum spatial resolutions change significantly. Figure 54(c) shows that only the top-down cloaking algorithm experiences an increase in cloaking time as the maximum spatial resolution values increase, while other cloaking algorithms are not very time-sensitive to maximum spatial resolution. Finally, Figure 54(d) shows that with increasing maximum spatial resolution, the relative spatial resolution (RSR) for all cloaking algorithms will increase proportionally, with a close to constant gap between BPE and the dynamic grid algorithms.

#### *6.8.2.4 Scalability*

We now study the scalability of the PRIVACYGRID system with respect to the changing number of mobile users. As the number of users in the system increases, we can expect the cloaking time for algorithms to generally decrease, since messages will be anonymized more easily, but the update costs for the grid-based structures will also increase. We use a similar setup to that in Section 6.8.2.3 with the mean spatial resolution fixed at 800 m with

5% standard deviation. We vary the number of users from 10K to 100K and observe the effect on the cloaking time and update cost. From Figure 55(a) we observe some interesting results. First, the amount of difference in cloaking time among the algorithms changes slightly with the increase in the number of mobile users. Second, TDDR shows a modest increase in cloaking time with the increase in the number of mobile users in the system. This is because the approach requires more iterations, since messages can be cloaked to smaller spatial regions. However, BUDE displays a reverse trend – the cloaking time decreases as the number of users increases. This is because a higher density of mobile users per grid cell will enable it to find the smallest cloaking box faster. Finally, we observe that HDER adapts well to the increase in the number of users, offering similar performance as BUDE in terms of cloaking time. Figure 55(b) measures the total number of updates per second required to update the grid-based data structures as the number of mobile users increases. For this experiment, the grid index is maintained as a main memory data structure. Each user provides a location update to the system after moving a distance of 100 m. We observe that BPE requires a large number of updates as the number of users increases. A nine-level pyramid is used in this experiment, requiring an average of 8 to 9 updates per location update request. In contrast, the dynamic cloaking approaches use the flat grid index, requiring only 2 updates for each location update request, which is significantly lower than the BPE approach [82].

#### *6.8.2.5 Distribution of User Location within Cloaked Areas*

This experiment is designed to study the distribution of the user location within the cloaking area. Figure 56 displays the user location distribution (ULD) for the different cloaking algorithms. Plotting the ULD allows us to observe the distribution of the normalized distance from the center of cloaking area to the actual user position for each of the algorithms. The more uniform the distribution of user locations is within the cloaking areas, the harder it is for an adversary to guess the actual location of the user within the cloaking area. We

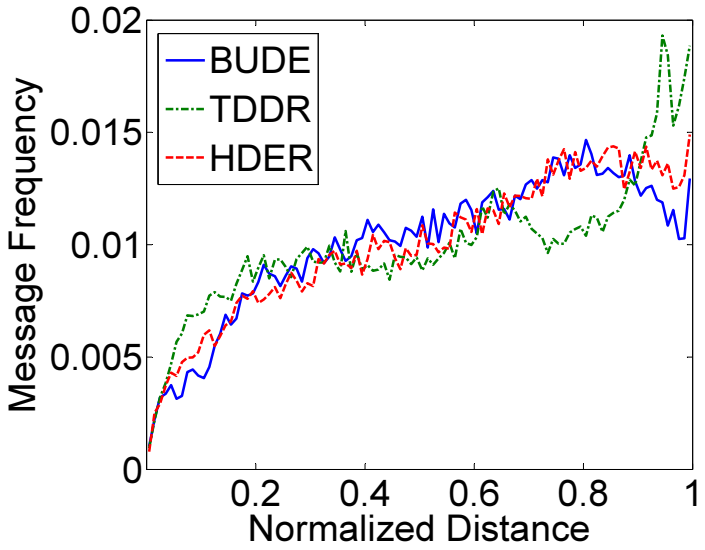



**Figure 56:** User Location Distribution

observe that all dynamic cloaking algorithms provide a rather uniform distribution of user location within the cloaking area. None of the approaches reveal any significantly skewed ULD patterns.

# CHAPTER VII

# MOBICLOAK: LOCATION ANONYMIZATION FOR MOBILE USERS ON ROAD NETWORKS WITH RECIPROCITY SUPPORT

## *7.1 Introduction*

The widespread availability of location sensing technologies has led to the successful deployment of a large number of location-based services and applications. The plethora of services available today provide a huge scope for business opportunities. However, the availability of continuous location information opens the door for potential misuse [64] as it can be used for stalking, performing inference about a user's medical conditions, or for location-based spam.

Location privacy refers to the capability of enabling a mobile node or a trusted server to conceal the relation between location and personal identifiable information from third parties. A fair amount of work has been performed to address the problem of location privacy and most existing research can be classified into spatial cloaking [28, 48, 53, 82, 51] and mix zone-based solutions [34, 54, 41, 46]. Most of the anonymization work focuses on privacy-utility trade-offs of the anonymized location information. However, this fails in both privacy protection and utility preservation for mobile users traveling on road networks.

The main problem arises due to the fact that the attained privacy level falls below the promised privacy level when the underlying road network is considered. As an example, we consider spatial cloaking techniques [28, 48, 53, 82] where the exact user position is anonymized to an area containing other users, thus rendering an adversary incapable of identifying the actual user position within the cloaked location. However, when the underlying road network is considered it may be possible for the adversary to identify that

the user is moving on a particular road segment. This unauthorized exposure of segment-identity relationship may have undesirable consequences. For example, if the road segment can be identified as the only road to enter a religious or a political club, then the probability of association of the mobile user with this sensitive public location can be intrusive in terms of privacy. Furthermore, utility of location information formulated using rectangular cloaking boxes becomes insufficient for services which use road network distance rather than Euclidean distance.

XStar [108] presents the first work of combining segment $l$-diversity with location $k$-anonymity using star-based cloaking, aiming at offering privacy aware mobile services over road networks. Concretely, by segment $l$-diversity definition introduced in [108], a subgraph of $l$ connected segments without a high degree forking junction can be a cloaked subgraph. It is obvious that such subgraphs often fail to preserve the true $l$-segment diversity since an adversary can easily link a request with a road segment in such a simple line graph of $l$ segments with higher probability than $1/l$. An extreme case is when these $l$ segments are small logical segments of an actual long road segment linking two star structures.

Another problem with XStar [108] is the lack of explicit support of the reciprocity criterion for successful cloaking. Informally, reciprocity states that by satisfying location $k$-anonymity, there should be at least $k$ users using the same cloaked location in their service requests. Unfortunately, many existing works [53, 82, 28] fail to meet the reciprocity requirement since their definition of location $k$-anonymity simply requires that there are $k - 1$ other users residing in the same cloaked location. As a result, instead of cloaking $k$ users with a unique cloaking region, each of the $k$ users is cloaked separately, resulting in different cloaking regions being reported by each of these $k$ users. The probability of linking a user with a request is much higher than $1/k$ when reciprocity is not met [51].

In this chapter, we present MOBICLOAK, a reciprocity preserving road network-aware location privacy model for users traveling on road networks. To address the problems in existing solutions, MOBICLOAK defines segment $s$-anonymity as a companion metric to the

user $k$-anonymity metric. Our $k$-anonymity definition explicitly addresses the reciprocity requirement whereas segment $s$-anonymity mandates forking junctions as a necessary constraint to strengthen privacy measures. The MOBICLOAK approach makes three unique contributions. First, we argue that the graph density of a cloaked subgraph is an important quality measure for road network-based location privacy protection. High cloaking graph density leads to high utility in terms of spatial resolution in the road network. Compact subgraphs lead to lower query processing costs based on current road network-based query evaluation cost models [88].

Second, we present graph-based cloaking algorithms, offering different levels of optimization in terms of privacy-utility trade-offs. Concretely, we use *randomized expansion* and *network expansion* as two naïve baseline techniques, each of which represents one end of the privacy-utility spectrum. As opposed to the work presented in [108], our algorithms ensure reciprocity of requests which are anonymized together. The randomized expansion technique, although it displays high attack resilience, suffers from poor utility. In order to deal with the static nature of the network expansion technique, which leads to low attack resilience, we introduce service request density-aware techniques, which utilize additional controlled randomness to provide higher resilience to replay attacks, while maintaining desired graph density of cloaked location.

Last but not the least, MOBICLOAK permits deferred cloaking as a mechanism to allow for trade-offs between spatial and temporal resolution of cloaking requests. We evaluate the performance of MOBICLOAK anonymization algorithms through analytical modeling and experimental evaluation.

## *7.2 Network-Aware Privacy Model*

This section lays the groundwork for our road network-aware location cloaking algorithm development. We formally introduce the MOBICLOAK location privacy model, the anonymization procedure, and the evaluation metrics to study the effectiveness of our approach.

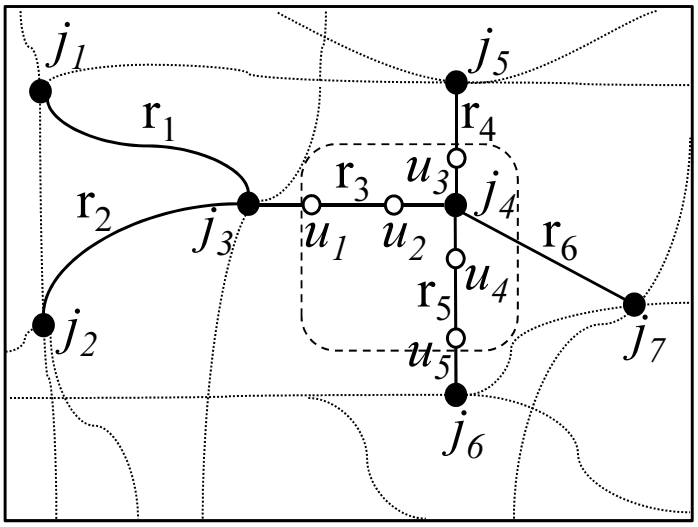



**Figure 57:** Road Network-based Model

### 7.2.1 MobiCloak Location Privacy Model

In MOBICLOAK, each mobile user has a personalized privacy preference profile (P3P) [28]. Two measures are used to capture the user-level location privacy requirements: user $k$-anonymity and segment $s$-anonymity. The first requirement ensures that the probability of an adversary being able to link a mobile user with a request generated from the cloaked subgraph is no greater than $1/k$. The companion requirement of segment $s$-anonymity ensures that the probability of an adversary being able to link a request with any road segment in the cloaked subgraph is no greater than $1/s$. This addresses the vulnerability of associating a user with a particular segment, which can be used to link the user to a sensitive public location such as some religious club or AIDS treatment center.

User $k$-anonymity cannot be guaranteed without ensuring segment $s$-anonymity in the road network-based model. For example, in Figure 57 a request from user $u_1$ with privacy parameters $k = 5$ and $s = 3$ is anonymized to include segments $r_3$, $r_4$ and $r_5$.

As shown by the dotted bounding region, five users $\{u_i\}_{i=1}^{5}$ are present within the region on the road segments. As long as the adversary is unable to associate a road segment with the service request, the probability of associating any user with the request is no greater than $1/5$. However, if the adversary is able to associate the request with a particular segment,

say $r_3$ in the figure, the user $k$-anonymity effectively falls to two. With these observations in mind, we argue that an anonymized request must satisfy the dual requirements of user anonymity and segment anonymity.

**Definition 7.2.1** *(User k-Anonymity) A user location is said to be $k$-anonymous if there are $(k-1)$ other users within the same location released by the anonymization server.*

**Definition 7.2.2** *(Segment s-Anonymity) The cloaked location of a user request is said to be segment $s$-anonymous if it contains at least $s$ distinct road segments, including the road segment associated with the user's actual location, as well as at least one forking junction of degree higher than two.*

Our definition of segment $s$-anonymity removes the possibility of producing a successful cloaked line subgraph. To further strengthen the attack resilience, we incorporate graph-density metric in both our cloaking algorithms for selecting the best candidate segment to expand and the evaluation of MOBICLOAK.

### 7.2.2 Location Anonymization Procedure

MOBICLOAK can be used by a trusted third party anonymizer to mediate between mobile users and LBS providers. We assume that given the current location of a mobile user, the anonymization server can determine the road segment associated with any generated service requests based on the underlying road network topology. All service requests issued by mobile users are forwarded to the anonymization server and represented by a message of the form $m_s = \langle obj_{id}, req_{id}, \{r, t\}, k, s, \sigma_s, \sigma_t \rangle$. The first two components of the message $- obj_{id}$ and $req_{id}$ uniquely identify a request. The spatio-temporal location of the message $m_s$ is represented by the road segment associated with the current user location $r$ and the timestamp $t$. The four parameters $\{k, s, \sigma_s, \sigma_t\}$ denote the P3P for the user issuing the service request. $\sigma_s$ denotes the spatial tolerance of the request which affects the size of the candidate result set [28] and $\sigma_t$ denotes the service delay which the user is willing to accept; together they constitute the utility metrics of the service request.

The general procedure of location anonymization is to take a service request $m_s$ as input and apply a specific road network-aware cloaking algorithm to generate a perturbed message $m_t = \langle h(obj_{id}||req_{id}), \mathcal{S} \rangle$, where $h$ is a secure hash function, $\mathcal{S}$ denotes the cloaked subgraph comprising at least $s$ road segments, including $r$. Additionally, the cloaked subgraph $\mathcal{S}$ must meet the *reciprocity* requirement. We formally define the reciprocity criterion using the notation of MOBICLOAK privacy model.

**Definition 7.2.3** *(Reciprocity) A cloaked subgraph $\mathcal{S}$ is said to meet the reciprocity requirement, iff $\mathcal{S}$ contains a set of service requests $M$ such that (i) $|M| \geq max_{i=1}^{|M|} m_i.k$, (ii) $|\mathcal{S}| \geq max_{i=1}^{|M|} m_i.s$, (iii) $\forall\, m_s \in M$, $\mathcal{S}$ is a subgraph of $B_{max}(m_i.\sigma_s, m_i.\sigma_t)$.*

The cloaked subgraph $\mathcal{S}$ lies within the *maximal spatio-temporal cloaking box* $B_{max}(m_i.\sigma_s, m_i.\sigma_t)$ of each message $m_i \in M$. Thus, an optimal algorithm would partition the road network graph into *reciprocity conformant* subgraphs which have minimal spatial resolution. However, optimal $k$-anonymity on its own is an NP-Hard problem [81]. In MOBICLOAK, the key idea is to provide controlled randomness when we construct a cloaking subgraph by expanding from the current segment on which a request resides. Concretely, to facilitate our implementation of reciprocity, we associate a segment profile $\kappa_r$ with each road segment $r$ as the basic data structure for MOBICLOAK algorithms to make an informed decision of segment expansion at each algorithmic step.

**Definition 7.2.4** *(Segment Profile $\kappa_r$) Each road segment $r$ is associated with a segment profile $\kappa_r = \langle M_r, max_{i=1}^{|M_r|} m_i.k, max_{i=1}^{|M_r|} m_i.s, min_{i=1}^{|M_r|} m_i.\sigma_s, min_{i=1}^{|M_r|} m_i.\sigma_t \rangle$, where $M_r$ denotes the set of messages associated with $r$.*

### 7.2.3 Evaluation Metrics

In this section, we define three sets of metrics that will be used to evaluate the cloaking algorithms. The first set of metrics is used to evaluate the level of privacy protection, and includes relative $k$-anonymity ($k_{rel}$), relative $s$-anonymity ($s_{rel}$), and segment entropy ($\mathcal{H}(\mathcal{S})$). The second set of metrics is used for evaluating the level of utility preserved in the

cloaked subgraph produced by a cloaking algorithm. Important utility measures include relative spatial resolution ($\sigma_{srel}(\mathcal{S})$), relative temporal resolution ($\sigma_{trel}(\mathcal{S})$) and graph density $\rho(\mathcal{S})$. The third set of metrics, anonymization success rate ($R$) and anonymization time $t$, is used for evaluating the cloaking algorithm performance.

**Relative k-anonymity ($k_{rel}$) and Relative s-anonymity ($s_{rel}$):** These two metrics measure the achieved levels of location $k$-anonymity and segment $s$-anonymity for successfully cloaked messages. Given a set of messages $M$ cloaked together, we define $k_{rel} = \frac{1}{|M|}\sum_{m_s \in M}\frac{|M|}{m_s.k}$. Similarly, $s_{rel} = \frac{1}{|M|}\sum_{m_s \in M}\frac{|S|}{m_s.s}$. Although our cloaking algorithms aim at obtaining higher anonymity, high anonymity achieved at the cost of a larger cloaked subgraph leads to higher processing cost of service requests.

**Segment Entropy ($\mathcal{H}(\mathcal{S})$):** The segment entropy is a measure of the privacy achieved by our cloaking techniques. We measure segment entropy as $\mathcal{H}(\mathcal{S}) = -\sum_{i=1}^{|\mathcal{S}|} p_i.\log p_i$, where $p_i$ denotes the probability of the user position being associated with the $i^{th}$ segment in $\mathcal{S}$.

**Relative Spatial Resolution ($\sigma_{srel}(\mathcal{S})$):** This metric measures the ability of a cloaking algorithm to provide compact subgraphs. Concretely, we define the $\sigma_{srel}(\mathcal{S})$ over a set of messages $M$ by $\frac{1}{|M|}\sum_{m_s \in M}\frac{m_s.\sigma_s}{\sigma(\mathcal{S})}$, where $\sigma(\mathcal{S})$ represents the spatial resolution (radius) of the cloaked subgraph $\mathcal{S}$.

**Relative Temporal Resolution ($\sigma_{trel}(\mathcal{S})$):** This metric measures the ratio of the maximum allowable delay $\sigma_t$ over the amount of delay introduced by a perturbed message $m_t$. Let $I = [t_s, t_e]$ denote the time interval between the earliest and latest message $\in M$. We define $\sigma_{trel}(\mathcal{S})$ over a set of messages $M$ by $\frac{1}{|M|}\sum_{m_s \in M}\frac{m_s.\sigma_t}{|m_t.I|}$.

**Average Graph Density ($\rho(\mathcal{S})$):** The average graph density is a measure of the compactness of the cloaked subgraph. The graph density for a cloaked subgraph is defined as $\rho(\mathcal{S}) = \frac{2.|E|}{|V|.|V-1|}$ where $|V|, |E|$ denote the number of nodes, edges respectively in the cloaked subgraph.

**Anonymization Success Rate ($R$):** Let $M$ denote the set of anonymization requests received by the system. The set of messages that are successfully perturbed can be computed

by $\{m_t | m_t = g_{cloak}(m_s),\ m_s \in M\}$, where $g_{cloak}(m_s)$ denotes an anonymization algorithm. The success rate is defined as follows: $R(g_{cloak}(m_s)) = \frac{|\{m_t | m_t = g_{cloak}(m_s), m_s \in M\}|}{|M|}$.

**Message Anonymization Time ($t$):** This metric measures the run-time performance of the cloaking algorithm in terms of time complexity. Efficient cloaking implies that the cloaking algorithm spends less time to perturb more messages.

## *7.3 Anonymization Algorithms*

In this section, we present two sets of graph-based cloaking algorithms. Each algorithm accepts as input a message $m_s$ for a service request. Neighboring segments in the road network are selected starting from the segment associated with $m_s$. The segment selection process is iterative and each step aims at meeting the privacy constraints $k$ and $s$ without violating the utility constraints $\sigma_s$ and $\sigma_t$ for all requests associated with the set of selected segments. The efficiency of a cloaking algorithm is determined by its segment selection heuristic.

### 7.3.1 Naïve Baseline Cloaking Algorithms

The first set of cloaking algorithms serve as naïve baseline methods and includes basic network expansion and randomized expansion. Both algorithms make the segment expansion decision solely based on the underlying road network topology. We now illustrate their inability to provide a good balance between privacy and utility of cloaked locations.

**Network Expansion:** Network expansion is designed using a variation of Djikstra's shortest path algorithm. The procedure starts by initializing the cloaked subgraph with the segment associated with the initial service request. Segment profile $\kappa_r$ is used to determine the current privacy and utility constraints. The procedure iterates the segment expansion process by first checking if the current subgraph satisfies the privacy requirements. If not, the algorithm selects the road segment that is closest to the user's current position in terms of road network travel distance [88] and satisfies the reciprocity requirement. The selected segment is added to the intermediate cloaking subgraph, and anonymization parameters are

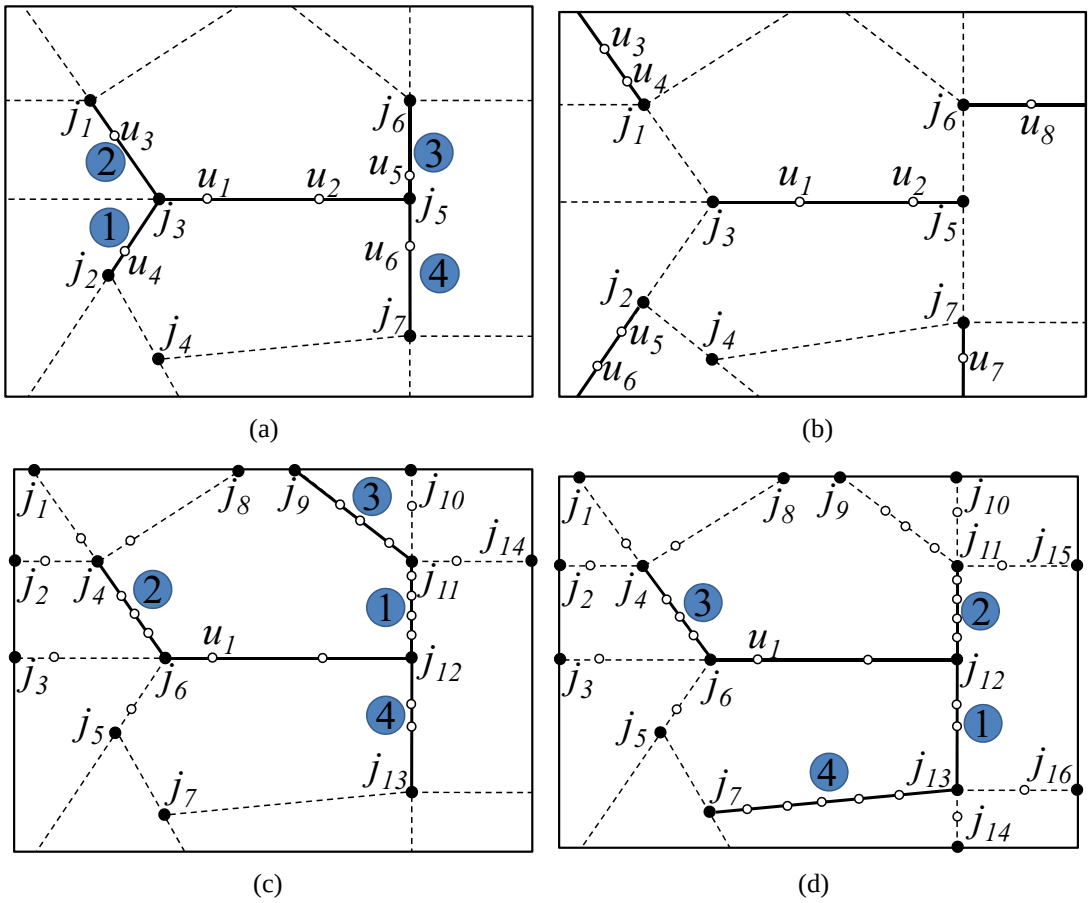



**Figure 58:** (a) Network Expansion (b) Randomized Expansion (c) Density-Aware Expansion (d) Density-Aware Randomized Expansion

updated with the profile of the newly added segment. If the current subgraph meets the reciprocity requirements, the algorithm terminates successfully. Otherwise, the iterative procedure continues till the reciprocity requirements are met (success) or no new segment can be added to the current subgraph (failure). The cloaked subgraphs generated are compact with small spatial extent and high graph density. However, this approach suffers from poor attack resilience since it uses a deterministic expansion heuristic. Figure 58(a) displays an example anonymization using this approach with the order of addition of segments denoted by the encircled numbers.

**Randomized Expansion:** Randomized expansion computes a segment pool comprising of segments with profiles satisfying maximal spatial and temporal reciprocity requirements. Then, starting with the initial segment, it randomly selects a segment from the segment

pool to add to the cloaked subgraph. For example, Figure 58(b) shows the same example fragment of the road network where user $u_1$ issues a request. The profile of segment $\overline{j_3 j_5}$, has $(k, s) = (5, 3)$. However, request from user $u_8$ has $s = 5$ which modifies the overall privacy requirements. The algorithm terminates when the cloaked subgraph meets the reciprocity criteria for privacy requirements. Randomized expansion allows for disconnected segments, which results in less compact subgraphs, and high service processing costs. However, using a set of disconnected segments as the cloaked location offers high resilience to replay attacks (see Section 7.4).

In summary, the use of the underlying road network topology in the network expansion algorithm makes the segment selection static and deterministic in nature and thus results in low attack resilience. On the other hand, randomized segment expansion leads to low graph density, though it offers high attack resilience.

### 7.3.2 Service Request Density-Aware Cloaking

In order to overcome the inability of naïve approaches to balance privacy-utility trade-offs, we develop density-aware dynamic cloaking techniques. The main idea is to utilize the segment profiles (mainly $|M_r|$) associated with each segment, which is dynamic in nature, to determine which segment should be added next to the cloaked subgraph. Given that the request count $|M_r|$ is highly dynamic, an adversary with knowledge of the underlying road network is unable to perform replay attacks successfully. We show that density-aware expansion can generate subgraphs with higher graph density; yet it is more resilient than naïve network expansion. Density-aware randomized expansion introduces controlled randomness in the cloaking process in order to strengthen the attack resilience against adversarial background knowledge.

**Density-Aware Dynamic Expansion:** Density-aware dynamic expansion takes a service request $m_s$ and starts with the initial cloaking subgraph $\mathcal{S}$ containing only the segment

$r$ associated with $m_s$. Two priority queues are created and maintained: segment count-based node priority queue $Q_n$ and request count-based segment priority queue $Q_s$. The two end nodes of $r$ are used to initialize $Q_n$ and are inserted in the decreasing order of the segment count connected to the node. The algorithm then inserts all segments connected to these nodes which satisfy the spatial and temporal reciprocity requirements into $Q_s$ in decreasing order of their request count. The segment at top of $Q_s$ is added to the cloaked subgraph and the anonymization profile is updated based on the newly added segment. Furthermore, the other end node of this segment is added to $Q_n$. All segments connected to this node junction, satisfying the reciprocity requirement, are inserted into $Q_s$. This procedure is repeated till either a cloaked subgraph which meets the reciprocity criteria is found (success) or no new segment can be added (failure).

Consider the example in Figure 58(c) where user $u_1$ issues a request with $(k, s) = (5, 5)$. For simplicity, we assume that other requests in the neighborhood have lower privacy requirements and do not violate spatial and temporal reciprocity requirements. The segment $\overline{j_6 j_{12}}$ is added to the cloaking subgraph. The algorithm next inserts the segments $\overline{j_3 j_6}$, $\overline{j_4 j_6}$, $\overline{j_5 j_6}$, $\overline{j_{11} j_{12}}$, $\overline{j_{12} j_{13}}$ connected to junctions $j_6$ and $j_{12}$ into $Q_s$ in the decreasing order of their request count. Segment $\overline{j_{11} j_{12}}$ has the highest request count of four and is selected to add to the cloaking subgraph which currently contains the segment $\overline{j_6 j_{12}}$ only. At this point, the $k$-anonymity requirements are met but the $s$-anonymity requirement is not satisfied. Thus the algorithm continues to expand the cloaking subgraph. The procedure is repeated till the cloaking subgraph meets the segment $s$-anonymity requirement.

This approach leads to much higher $k$-anonymity ($k = 14$ in the above example) compared to network expansion approach, thus providing higher privacy with similar QoS. Furthermore, unlike network expansion, density-aware expansion uses request counts, which are dynamic in nature, as an expansion criterion making it harder for an adversary to mount an attack. However, if an adversary could continuously log requests on the road network, it is possible to perform a replay attack.

**Density-Aware Dynamic Randomized Expansion:** In order to overcome the deficiency of density-aware technique, we design an algorithm that can break the deterministic nature of the request count-based expansion. Introduction of controlled randomness makes the generated subgraphs highly attack resilient.

Concretely, this algorithm requires a modification to the previous approach. While selecting a segment to add to the cloaking subgraph the algorithm does not select the segment at the top of $Q_s$. It considers all segments with request counts in the range of $[\lceil |M_r^{top}| * (1-\alpha) \rceil, |M_r^{top}|]$ and randomly selects a segment to add to the cloaked subgraph from this subset. $|M_r^{top}|$ represents the request count associated with the segment at the top of $Q_s$ and $\alpha$ represents the randomization factor supplied by the system. This approach makes it impossible for the adversary to launch a replay attack even in the presence of complete knowledge of the segment profiles and service request positions (Section 7.4).

We illustrate this algorithm through Figure 58(d), where user $u_1$ performs a request with $k = 5$ and $s = 5$. The request is anonymized using the randomization factor $\alpha = 0.5$. The request lies on the segment $\overline{j_6 j_{12}}$. The algorithm next inserts segments $\overline{j_3 j_6}$, $\overline{j_4 j_6}$, $\overline{j_5 j_6}$, $\overline{j_{11} j_{12}}$, $\overline{j_{12} j_{13}}$ connected to junctions $j_6$ and $j_{12}$ into $Q_s$ in order of their request count. Segment $\overline{j_{11} j_{12}}$, with a request count of four, lies at the head of $Q_s$. All three segments with mobile object counts in the range of [2, 4] are considered while expanding the cloaking subgraph. The algorithm may randomly select any of the three segments to extend the current cloaked subgraph. Figure 58(d) shows the algorithm selects $\overline{j_{12} j_{13}}$ as the next segment and proceeds in an iterative manner. The effect of the randomization factor is studied in the next section.

## *7.4 Threat Model and Analysis*

In this section, we outline an attack method which may be used by an adversary to perform a replay attack on the road network-based location anonymization solutions. We argue that the resilience of the anonymization algorithms against this threat model determines the

success of the solution in protecting location privacy of mobile users traveling on road networks. We consider the attack resilience of the algorithms against ad-hoc service requests because we assume that all location services considered in our context do not require identity or pseudo-identity. Thus, each evaluation of the service request is considered as an independent message in our location anonymization process. There is no pseudo-identity based association for successive requests generated by the same user.

The anonymization algorithm generates a set of segments $\mathcal{S}$ as the cloaked subgraph. Ideally, an adversary is able to identify that a user belongs to a segment $s \in \mathcal{S}$, with a probability no greater than $1/|\mathcal{S}|$. In this scenario, each segment is indistinguishable to the adversary from all other segments in the cloaked subgraph $\mathcal{S}$. Our attacker-centric threat model assumes that given a cloaked subgraph $\mathcal{S}$ and knowledge $\mathcal{K}$, the adversary attempts to determine the probability of each segment $s \in \mathcal{S}$ being the actual segment associated with the user location. The knowledge $\mathcal{K}$ of the adversary considered in this work includes: 1) Knowledge $\mathcal{K}_r$ of the underlying road network, 2) Knowledge $\mathcal{K}_a$ of the various anonymization algorithms, and 3) Knowledge $\mathcal{K}_o$ of the mobile object counts on the various road segments. The adversary uses this information to conduct a replay attack in order to determine the linking relationship between segment and user identity.

### 7.4.1 Replay Attack

Given that the segment associated with the user generating the service request was $s'$, we denote the probability of a user identifying the $i^{th}$ segment in $\mathcal{S}$ as $s'$, $s_i = s'$, as $p_i$.

$$p_i = prob[s_i = s'|\mathcal{S}, \mathcal{K}]$$

For each segment $s_i \in \mathcal{S}$, the adversary attempts to *replay* the algorithm assuming that the segment $s_i$ is associated with the exact location of the user. The probability of the adversary generating $\mathcal{S}$ using the replay attack, given the assumption $s_i = s'$ is,

$$p_{i,\mathcal{S}} = prob[\mathcal{S}|s_i = s', \mathcal{K}].$$

Therefore, we get,

$$prob[s_i = s' | \mathcal{S}, \mathcal{K}] = \frac{prob[\mathcal{S} | s_i = s', \mathcal{K}]}{\sum_{i=1}^{|\mathcal{S}|} prob[\mathcal{S} | s_i = s', \mathcal{K}]}$$

For each algorithm, the adversary generates all the possible sets of segments $\mathcal{S}_k$, such that $|\mathcal{S}| = |\mathcal{S}_k|$ , $k = [1...n]$. Then,

$$prob[\mathcal{S} | s_i = s', \mathcal{K}] = \sum_{j=1}^{k} \frac{b_j}{n},$$

where $b_j = 1$ if $\mathcal{S}_k = \mathcal{S}$, else $b_j = 0$.

### 7.4.2 Algorithm Analysis

Now we analyze each of the algorithms to determine the probability of an adversary identifying the $i^{th}$ segment $s_i$ to be associated with the exact user location.

**Randomized Expansion:** Randomized expansion selects the $(|\mathcal{S} - 1|)$ segments at random and adds them to the cloaked subgraph comprising of the segment associated with the actual user location. The replay attack model will determine that the likelihood of each segment being identified as $s'$ is equal, i.e.

$$prob[s_i = s' | \mathcal{S}, \mathcal{K}_r, \mathcal{K}_a] = \frac{1}{|\mathcal{S}|}, \forall i \in [1...|\mathcal{S}|].$$

**Network Expansion:** Under this model, the $(|\mathcal{S}| - 1)$ extra segments are produced using a deterministic algorithm. The actual segment $s'$ associated with the exact user location is very likely to generate $\mathcal{S}$ as the subgraph. The generated subgraph may not be equal to $\mathcal{S}$ considering the point of the segment used for generating the subgraph. Other segments may or may not generate the subgraph $\mathcal{S}$. This approach is likely to have low entropy value as verified by our experiments. The knowledge available to the adversary is assumed to be $\mathcal{K}_r$ and $\mathcal{K}_a$.

**Density-Aware Dynamic Expansion:** Once again, the $(|\mathcal{S}| - 1)$ extra segments are produced using a deterministic algorithm. The only randomness arises in tie-break situations where more than one segment is present at the top of the priority queue with the same

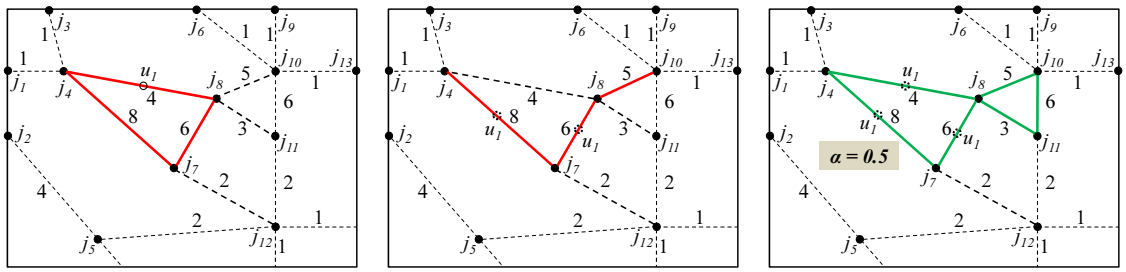



**Figure 59:** Possible Attack on Mobility-Aware Dynamic Network Expansion and Removal of Vulnerability by Introduction of Controlled Randomness

number of mobile objects. The actual segment $s'$ associated with the exact user location will generate $\mathcal{S}$ as one of the possible subgraphs. Other segments may or may not generate the subgraph $\mathcal{S}$. This approach is likely to have low entropy value as verified by our experiments. The knowledge available to the adversary is assumed to be $\mathcal{K}_r$, $\mathcal{K}_a$ and $\mathcal{K}_o$.

**Density-Aware Dynamic Randomized Expansion:** This approach is likely to generate lots of possible subgraphs for each segment in $\mathcal{S}$ dependent on the value of the randomness factor $\alpha$. Higher the value of $\alpha$, larger the number of possible segments which may be added to the current cloaked subgraph. Thus, $\alpha$ controls the number of possible subgraphs which may be generated using a particular segment $s_i$ as the starting point. Higher the value of $\alpha$, lower the probability of an adversary being able to identify the segment associated with the exact user location. The knowledge available to the adversary is assumed to be $\mathcal{K}_r$, $\mathcal{K}_a$ and $\mathcal{K}_o$.

Figure 59 displays an example attack on the mobile object count based network expansion algorithm. In Figure 59(a), user $u_1$ on segment $\overline{j_4 j_8}$ issues a LBS request with $(k, s) = (15, 3)$. Assume the algorithm generates the cloaked subgraph comprising of the three segments $\overline{j_4 j_8}$, $\overline{j_4 j_7}$ and $\overline{j_7 j_8}$, as displayed in the figure in solid lines forming a triangle. An adversary using the replay attack can deterministically conclude that the exact location of $u_1$ lies on $\overline{j_4 j_8}$. This is because if the exact location of $u_1$ was on either segment $\overline{j_4 j_7}$ or segment $\overline{j_7 j_8}$, the algorithm should result in a different cloaking subgraph as shown in Figure 59(b). In contrast, if we use the randomized mobility-aware network expansion

algorithm with the randomization factor set to $\alpha = 0.5$, the cloaked subgraph generated by this algorithm is displayed in bold lines shown in Figure 59(c). Clearly, every anonymization execution will possibly generate a different cloaked subgraph irrespective of which of the three segments the user position is being associated with.

## *7.5 Implementation Enhancements*

In *immediate cloaking*, each request is handled by the anonymization server at the time of arrival. We implement two enhancements on top of the algorithms to improve the performance of the system: *Early Failure Detection* and *Deferred Cloaking*. Early failure detection caches statistical information of neighboring segments to determine if a cloaking procedure is bound to fail due to insufficient number of requests in the surrounding network graph. Deferred cloaking delays the anonymization process in the absence of a sufficient number of requests (a factor $\lambda \times \kappa_r.k$, $\lambda \geq 1$) within the spatial tolerance limits. Temporal tolerance associated with the request allows delay in cloaking, thus presenting opportunities to trade-off the achieved spatial and temporal resolution. As claimed in other location privacy work [108], *shared processing of multiple queries* is a further enhancement which may be implemented. Note that in MOBICLOAK, the principle of reciprocity naturally enables shared processing of multiple queries for all requests associated with the corresponding cloaked subgraph.

## *7.6 Experimental Evaluation*

In this section, we perform an empirical analysis of the algorithms - Network Expansion (NE), Randomized Expansion (RE), Density-Aware Dynamic Expansion (DAE) and Density-Aware Dynamic Randomized Expansion (DARE) - based on the metrics defined in Section 7.2.3. The experimental evaluation is focused on the effectiveness of our cloaking algorithms in terms of privacy and QoS metrics, entropy-based privacy measure and performance of the anonymization algorithms.

### 7.6.1 Experimental Setup

Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey. Vehicles are uniformly placed on the road network according to traffic densities determined from the traffic volume data in [53]. The simulator simulates the motion of 20,000 users on the road network with appropriate velocity information.

We use maps of varying sizes to observe the performance of the MOBICLOAK algorithms under different conditions. It is observed that the relative performance of the algorithms is similar under different size of road network topologies tested. Results reported in this chapter are measured using a map of Chamblee region of Georgia, which covers an area around 168 $km^2$ in expanse, to generate the trace. This work uses the gtmobisim simulator developed at Georgia Institute of Technology for generating traces of mobile users moving on a real world road network [90]. Each user is placed randomly at a source location based on the traffic volume data. The simulator uses shortest path routing to direct each user from its source location to its destination location. Location updates are generated for each mobile user containing the current time instant or timestamp associated with the location update. The user location is expressed as the current segment on which the user is located along with the distance from the starting point of the segment. The velocity of the user at this time instant is also recorded; further updates are only recorded for time instants where the user velocity experiences a change. This allows for generation of updates associated with the user location at any time instant. We use this velocity-based trace to generate periodic updates for mobile user location which are then fed to the location anonymization module. This setup follows the assumption that periodic location updates are generated by each mobile user and sent to the location service provider; requests are generated at a mean interval of 30 seconds.

The map comprises of about 10,000 road segments and 7,000 road junctions. Our experiments use traces generated by simulating vehicle movement for a period of one hour,

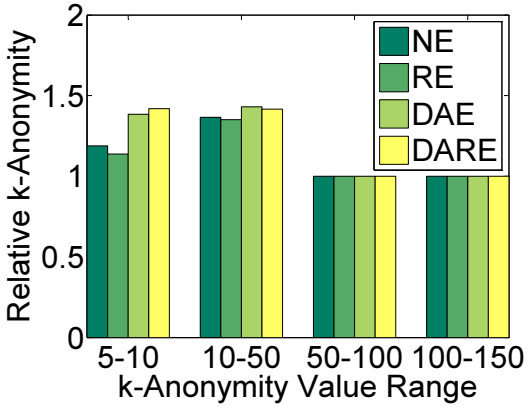
results are averaged over a number of such traces. Default $k$ values are chosen from the range $10 - 50$ and default $s$ values are chosen from the range $10 - 20$ along a Zipfian distribution with parameter value 0.6. Default values for $\sigma_s$ and $\sigma_t$ are $1 - 1.5$ km and $30$ seconds respectively. The default values for the randomization factor $\alpha$ and $\lambda$ are set to 0.5 and 2 respectively. Unless mentioned otherwise, parameters are set to their default values.

### 7.6.2 Experimental Results

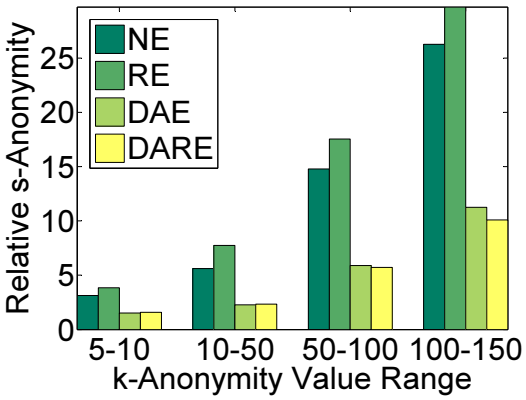
We first present the experimental results for user-defined privacy and QoS metrics, the entropy-based privacy metric, and the system performance by varying the values of $k$-anonymity. Our results show that (i) the density-aware techniques are more effective than naíve approaches on both privacy and QoS metrics, (ii) density-aware approaches may have slightly lower success rate as they are prone to selecting longer segments thus violating the $\sigma_s$ requirement in a few cases, (iii) DARE has attack resilience close to the RE approach, (iv) spatio-temporal cloaking allows for trade-offs between spatial and temporal resolution of cloaked messages.

#### *7.6.2.1 User-defined Privacy and QoS Metrics*

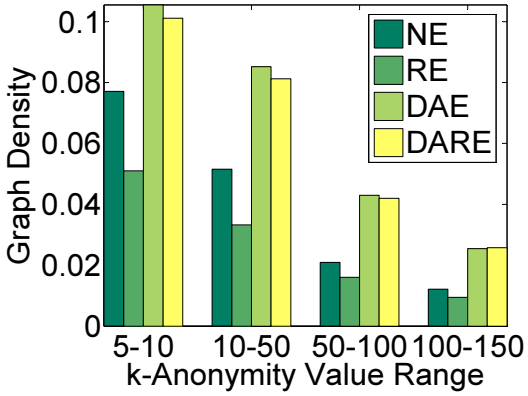
*Results with Varying $k$-Anonymity Levels:* We vary the $k$-anonymity values to test the performance of our algorithms for different privacy levels. We set the $s$-anonymity levels between $10 - 20$, values chosen along a Zipfian distribution with parameter $0.6$, and maximum temporal resolution to a mean value of $30$ seconds. The $k$-anonymity levels are chosen along a Zipfian distribution with parameter $0.6$ from the following set of range values: $5-10$, $10-50$, $50-100$ and $100-150$ with corresponding maximum spatial tolerance in the range $0.8 - 1$ km, $1 - 1.5$ km, $1.5 - 2$ km and $2 - 2.5$ km. In Figure 60(a), the relative $k$-anonymity values are higher for the density-aware approaches compared to naïve approaches for lower $k$-anonymity levels as the density-aware approaches select segments with higher request counts. For higher $k$-anonymity requirements, all approaches have relative $k$ levels close to one. However, as shown by the relative $s$ levels in Figure 60(b), the



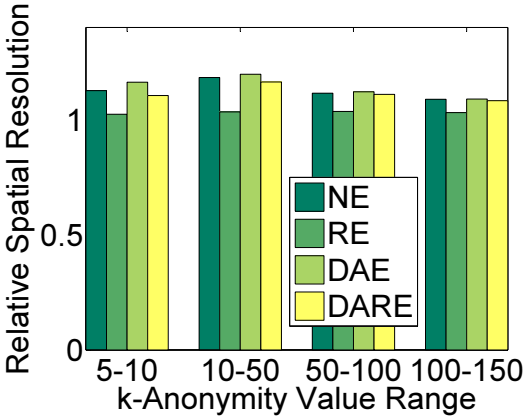

(a) Relative k-Anonymity Level



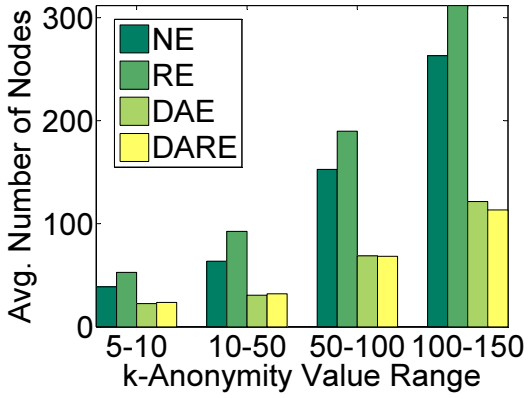

(b) Relative s-Anonymity Level




(c) Average Graph Density




(d) Relative Spatial Resolution




(e) Average Node Count

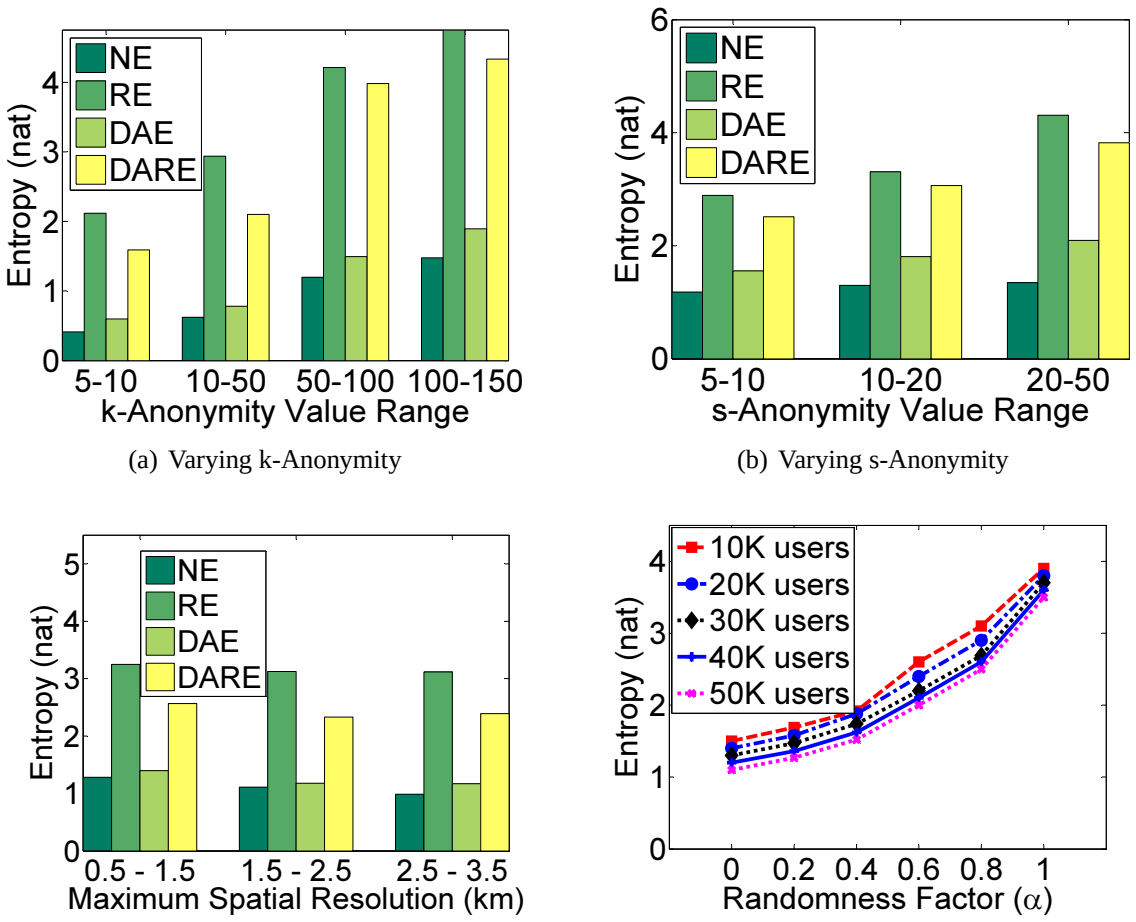
**Figure 60:** User-defined Privacy and QoS Metrics with Varying k-Anonymity Levels

density-aware approaches achieve higher $k$ levels by selecting fewer number of segments compared to the baseline approaches.

Figure 60(c) shows that density-aware approaches achieve consistently higher graph density compared to the randomized approach. Also note that the cloaking subgraphs produced by the density-aware schemes are compact enough to outperform the network expansion approach. As $k$ increases, the graph density achieved by all the approaches falls as larger subgraphs are produced. The relative spatial resolution of density-aware approaches is comparable to the network expansion approach(Figure 60(d)). We observe that longer road segments are expected to have larger number of requests and are typically selected for expansion with the density-aware approaches which impacts their spatial resolution negatively. Figure 60(e) displays the average node count of the cloaked subgraphs; again the density-aware approaches outperform the baseline approaches. Note that the overall query evaluation cost is dependent on the number of segments and nodes in the cloaked subgraph, which implies that density-aware approaches have the lowest query evaluation costs.

#### *7.6.2.2 Entropy-based Evaluation*

In this experiment, we measure the entropy levels achieved by each approach as described in Section 7.4.1. Figure 61(a) displays the entropy values with increasing $k$ values; $s$ values are chosen from the range $10-20$. As expected, the randomized expansion approach has the highest entropy levels, while the network expansion and the density-aware approach have relatively low entropy due to their deterministic nature. We see that introducing randomization to the density-aware approach results in much higher entropy levels; interestingly its entropy levels are competitive compared to the randomized approach. Figure 61(b) shows that entropy levels increase with increasing $s$ values ($k$ values in the range $10 - 50$) and density-aware randomized expansion has higher entropy values compared to the basic density-aware approach and network expansion. On the other hand, Figure 61(c) shows that as we increase the $\sigma_s$ values, the entropy levels remain more or less stable. This shows




(a) Varying k-Anonymity

(b) Varying s-Anonymity

(c) Varying Spatial Resolution

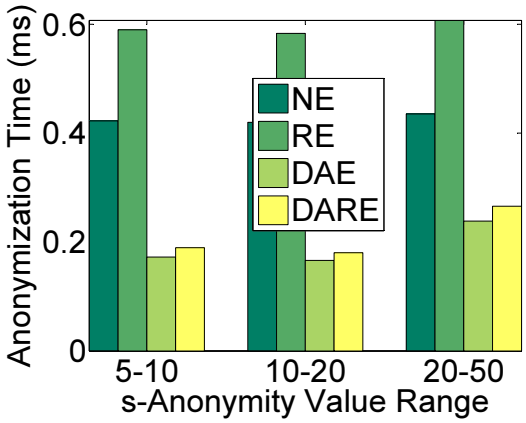
(d) Varying Randomness Factor

**Figure 61:** Entropy Values for the Anonymization Algorithms

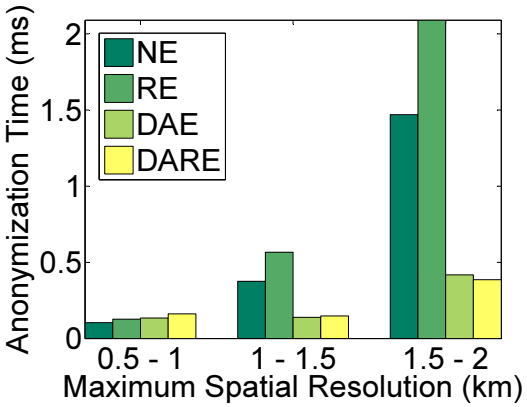
that it is hard for adversaries to launch a successful attack on our density-aware randomized expansion approach, irrespective of the desired spatial resolution. Figure 61(d) displays the entropy values as we increase the value of the randomization factor $\alpha$. As expected, entropy values increase sharply with increasing $\alpha$. Furthermore, the segment entropy decreases as we increase the number of users in the system. This is due to fewer number of segments being selected to meet the $k$-anonymity requirements due to higher request density.

#### *7.6.2.3 Performance Metrics*

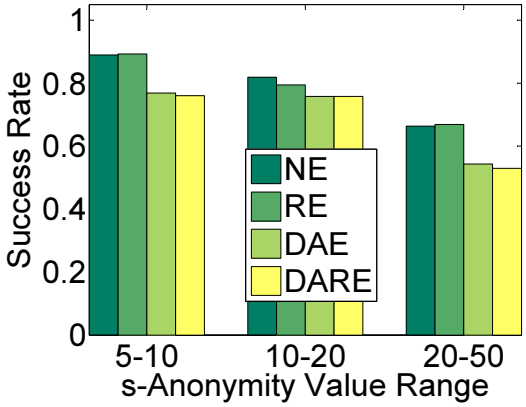
We observe two performance metrics for MOBICLOAK: anonymization success rate and average anonymization time with varying $s$-anonymity levels and varying $\sigma_s$. Figure 62(a)



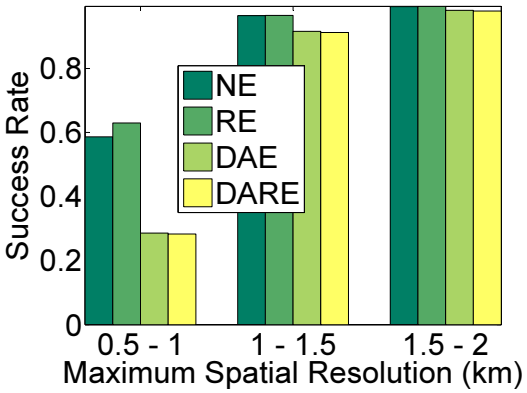

(a) Anonymization Time with Varying s-Anonymity




(b) Anonymization Time with Varying $\sigma_s$




(c) Success Rate with Varying s-Anonymity




(d) Success Rate with Varying $\sigma_s$

**Figure 62:** Performance Metrics with Varying s-Anonymity and Spatial Resolution $\sigma_s$

displays the average anonymization time measurements with increasing $s$ values. The density-aware approaches are faster as they anonymize requests quickly by locating denser subgraphs. Similar trends are observed for increasing $\sigma_s$ in Figure 62(b). The baseline randomized expansion approach requires more time to retrieve the segments in the pool and experiences high anonymization costs. The density-aware approaches have lower anonymization time, although it increases with increasing $\sigma_s$. Figure 62(c) displays that the success rate falls with increasing $s$ values. This is due to the fact that meeting the higher $s$ requirements with the same $\sigma_s$ values becomes more difficult. Figure 62(d) shows that the success rate increases with increasing $\sigma_s$ values as larger number of requests can be anonymized. In

addition, we also observe that the density-aware approaches have low success rate for very low spatial resolution values due to their tendency to select longer segments. However, for higher $\sigma_s$ values their success rate is easily comparable to the naïve approaches.

#### *7.6.2.4 Other Results*

*Effect of Varying $\sigma_t$* - This experiment is designed to study the effect of deferred cloaking for anonymization using the Density-Aware Randomized Expansion algorithm. We vary $\sigma_t$ with a mean value between $15 - 60$ seconds and the interval time for request generation with a mean value $15 - 60$ seconds too. The values of $\sigma_s$ are selected from the range $0.5 - 1$ km.

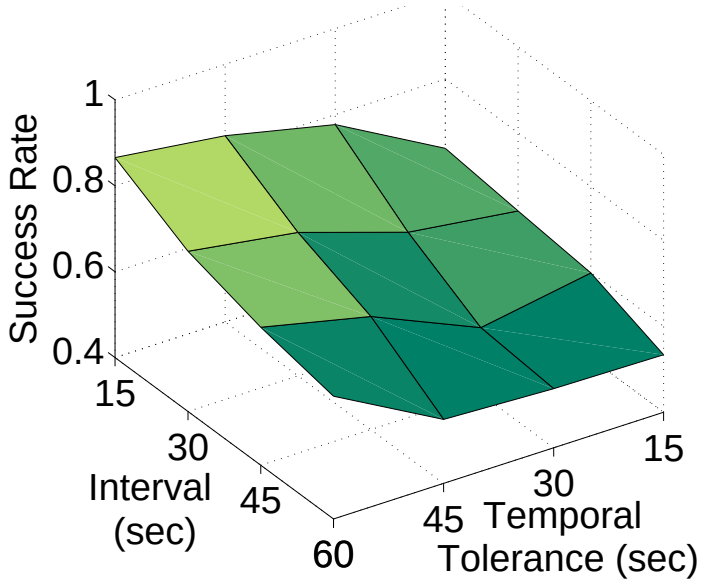
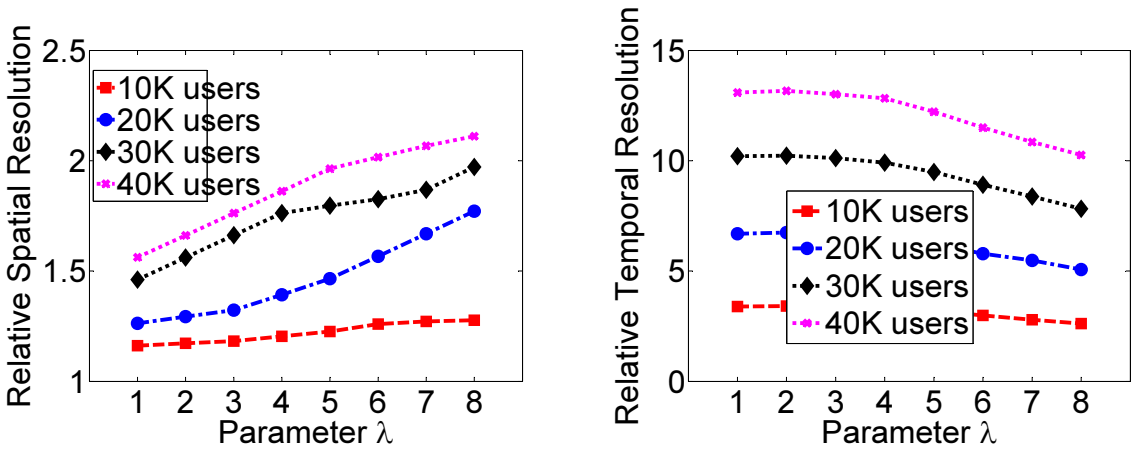



**Figure 63:** Varying $\sigma_t$ and Interval

Results are shown in Figure 63. The success rate increases with increase in temporal tolerance $\sigma_t$ as the messages are present in the system for a longer duration before they are dropped. Success rate also increases with a decrease in mean interval period as messages are generated more frequently leading to more cloaking opportunities.

*Effect of Deferred Cloaking* - We study the effect of deferred cloaking on the relative spatial and temporal resolution, by varying the values of parameter $\lambda$ in the range $1 - 8$ and the




(a) Relative Spatial Resolution with Varying $\lambda$ (b) Relative Temporal Resolution with Varying $\lambda$

**Figure 64:** Effect of Deferred Cloaking

number of users from 10,000 - 40,000, with the Density-Aware Randomized Expansion algorithm. Default values, as described earlier, were used for other parameters. As can be observed from Figure 64(a), the relative spatial resolution increases with increasing $\lambda$ as requests are cloaked in more compact subgraphs with higher density. The relative temporal resolution value falls with increasing $\lambda$ as requests are cloaked immediately only if a high density of other requests are present in the neighborhood (Figure 64(b)). Higher $\lambda$ values lead to a delay in cloaking and hence lower temporal resolution values.

## *7.7 Related Work*

Existing anonymization solutions utilize location perturbation as a mechanism to disable an adversary from associating personally identifiable information with a location. Representative techniques for anonymization-based solutions are spatial cloaking [28, 48, 51, 53, 82], false dummies [65] and landmark objects [56], with spatial cloaking being the most popular protection mechanism. The criterion of transformation is solely based on location anonymity, and the amount of protection measured in terms of the area of the cloaked region. Under the road network mobility model, the cloaking area is no longer an effective and valid measure.

There have been limited efforts on providing location privacy for users on road networks. XStar [108] is the first in-depth work that combines location $k$-anonymity with $l$-segment diversity, using star-based cloaking. XStar uses a road-network query processing cost model to trade-off between privacy and utility of the cloaked location. However, as we pointed out in Section 7.1, the $l$-segment diversity definition in [108] lacks graph density consideration and explicit support of the reciprocity criterion for successful cloaking. In fact, many existing location anonymization algorithms [53, 82, 28] fail to meet the reciprocity requirement. To the best of our knowledge, only CliqueCloak [48] and Prive [51] to date have made reciprocity as a mandatory criterion for successful cloaking.

Processing spatial queries over road networks has been an emerging research topic [88, 67, 36, 85]. The cost of query evaluation is a function of the set of segments and set of nodes considered for evaluation; thus, the MOBICLOAK approach is bound to have lowest query evaluation costs.

# CHAPTER VIII

# CONCLUSION AND FUTURE WORK

The availability of a large number of location-aware devices has enabled a new class of applications, known as location-based services (LBS), offering both new business opportunities and a wide array of new quality of life enhancing services. New applications such as location-based advertisement, location-based alerts or reminders, location-enhanced social networking and a host of other applications provide useful information based on the current location of mobile users. Despite advances in technology, a number of issues need to be considered for facilitating upcoming applications. Scalability of back-end servers is one important requirement in order to allow these services to host a large number of users. Secondly, despite the advances in mobile devices, battery life is a major issue to consider. Mobile devices operate in a wireless environment where energy and bandwidth consumption are important considerations. Resource-aware methods for facilitating applications will aid their acceptance and deployment. The ability to locate mobile users accurately also opens door for new threats, namely, the intrusion of location privacy. The availability of a large amount of location data may lead to unauthorized information exposure about the individuals' medical conditions, alternative lifestyles, unpopular political views or location-based spam and stalking.

This dissertation provided solutions to these important challenges for location-based service provisioning. Firstly, we described the spatial alarm framework as the basic primitive in order to provide building blocks for location-based services that require location-based trigger capability. Applications developed on top of this framework are enabled to use this generalized framework and a suite of optimization techniques for server-centric scalable processing. The spatial alarm framework provides an architecture and algorithms

for enhancing system scalability compared to naïve processing techniques. It also reduces communication costs and energy consumption on the client side. We presented a systematic approach to the design and development of the spatial alarm processing framework and various optimization techniques.

Secondly, we developed an architecture for privacy-enhanced location service provisioning, focused on providing customizable and personalized location privacy solutions for scalable mobile systems and services under random waypoint and road network mobility models. Our development of PRIVACYGRID and MOBICLOAK protects location privacy of mobile users while maintaining the end-to-end QoS for location-based service provisioning in the presence of dynamic and personalized privacy constraints.

We summarize the major developments of this dissertation in the next section and outline open issues and future directions in section 8.2.

## *8.1 Dissertation Conclusion*

We summarize below the major contributions of this thesis towards addressing the challenges associated with developing scalable, privacy enhanced location-based services and applications.

- We developed a location-centric indexing framework which advocates a clean separation of static objects from moving objects in terms of both spatial indexing structure and location query processing. By promoting locations as *first class citizens*, we displayed that the location indexing framework enables scaling of location-based services. Concretely, we build a location index for managing all the static objects in terms of their geographical locations in the real world. Our formal analysis and experimental evaluation both confirmed that our location indexing framework can provide fast access capability with low maintenance costs compared to the existing object indexing schemes.

- We developed the spatial alarms framework as a generalized framework for allowing

new location-based applications to express their location-based needs in the form of location-based triggers. We proposed the usage of this framework as a vital building block for a large number of upcoming location-based applications.

- We optimized the conventional approach of periodic alarm processing by advocating a motion-aware *safe period-based* alarm evaluation framework. Concretely, we introduced the concept of *safe period* to minimize the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system. We displayed that our safe period-based alarm evaluation techniques can significantly reduce the server load for spatial alarm processing compared to the periodic evaluation approach, while preserving the accuracy and timeliness of spatial alarms. Furthermore, we also developed alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, aimed at optimizing safe period computation at the server. The scalability and accuracy of our approach is validated using a road network simulator. We displayed that our proposed framework offers significant performance enhancements for the alarm processing server while maintaining high accuracy of spatial alarms.

- We proposed a distributed architecture and a suite of safe region techniques for scalable processing of spatial alarms [26]. We displayed that our safe region-based processing enables resource optimal distribution of partial alarm processing tasks from the server to the mobile clients. This implies that our distributed architecture minimizes unnecessary alarm evaluations at both server and mobile clients. We proposed three different safe region computation algorithms to explore the impact of size and shape of the safe region on network bandwidth, server load and client energy consumption. Our suite of safe region computation techniques allows us to analyze the impact of the size and shape of safe region on the client-server communication cost, server load and client energy consumption. The alternative methods for safe region

computation provide flexible support for mobile clients with heterogeneous capabilities in terms of CPU, network bandwidth and energy capacity.

- We presented a three-tier architecture to provide location privacy using a third party anonymizer service. We developed location cloaking algorithms which are fast and capable of keeping the perceived delays due to location anonymization to a minimum. We developed *dynamic* grid cloaking algorithms which achieve high anonymization success rate and efficiency in terms of both time complexity and maintenance costs. A hybrid approach that carefully combines the strengths of our different cloaking approaches to further reduce the average anonymization time was also developed.

- We allow each user to specify her privacy requirements in terms of privacy and utility constraints. In PRIVACYGRID, we use location $k$-anonymity and location $l$-diversity as two quantitative metrics to model the location privacy requirements of a mobile user. As QoS measures, we used *maximum spatial resolution* which allows mobile users to control the spatial resolution reduction within an acceptable QoS specific range. It can be changed or adjusted according to the type of location service, the time of day, month or year, and on a per message level. Similarly, the fourth measure is *maximum temporal resolution*, which controls the temporal delay acceptable for maintaining the desired QoS. Our algorithms use these constraints effectively to balance the privacy utility trade-offs.

- We also presented MOBICLOAK, a reciprocity preserving road network-aware location privacy model for users traveling on road networks which overcomes the weaknesses of the random waypoint model of the PRIVACYGRID approach. To address the problems in existing solutions, MOBICLOAK defines segment $s$-anonymity as a companion metric to the user $k$-anonymity metric. We displayed that our approach provides high entropy measures even after taking the underlying road network into consideration.

## *8.2 Open Issues and Future Research Directions*

This dissertation addresses some of the important challenges for development of future mobile systems and services. At the same time, it also draws attention towards a set of new challenges that need to be addressed by further work in this area. We discuss this set of open issues in this section.

### 8.2.1 Spatial Alarms Extensions

This dissertation has not only contributed to the system-level development and optimization of spatial alarms in large scale mobile systems, but also opened several new avenues for research in this area. We give a brief discussion below on a selection of such open issues.

**Spatial Alarm as Cloud-based Middleware Service:** Deployment of the spatial alarms framework in a cloud environment throws up interesting research challenges. With the current trends moving towards development of cloud-based, large-scale data centers, latency becomes a major issue for real-time, latency sensitive applications. This dissertation has dealt with the distribution of dynamic and hotspot oriented workloads among servers and mobile clients. Our experience shows that developing a truly distributed spatial alarms system that can scale to large number of subscribers with large number of spatial alarms is still a big challenge.

Another issue directly related to scaling spatial alarms is the capability of dealing with device heterogeneity and application programming interface (API) heterogeneity used to access spatial alarm processing middleware servers hosted in a cloud environment. Currently, users in developed countries have a large variety of smart phones, equipped with powerful computational and communication capabilities. However, a large percentage of cellular phone users in many developing nations like India do not have access to such devices which provide a rich user experience. As a result, the development of mobile services and systems middleware hosted in a cloud infrastructure should take into consideration both device heterogeneity and application programming interface (API) heterogeneity in order

to deliver the convenience and benefits of location-based applications and services to the world wide population on a 24 by 7 basis. Middleware support for ease of development of such applications is a vital component at the intersection of mobile computing research and cloud computing research, which demands serious attention from both academia and IT industry.

**Extensions to Spatial Alarm Processing Algorithms:** The spatial alarm processing algorithms developed in this dissertation handle either static alarm target with moving subscribers or moving alarm target with static subscribers. In the latter case, we use the same algorithms but with alarm check set over moving alarm targets instead of mobile subscribers. The third type of spatial alarms are those with moving mobile subscribers and moving targets. Further extensions to the spatial alarm processing algorithms are required for supporting spatial alarms with moving subscribers and moving targets. Examples of applications which fall in this category include social networking applications like Google Latitude where one user may set a spatial alarm on another friend. The techniques developed in this dissertation support efficient alarm processing with 100% success rate when either static targets or static subscribers are considered. Techniques like the TPR$^*$-tree [103] can be extended for indexing of alarms which involve both moving subscriber and moving alarm target. In such algorithms, one needs to consider new ways to define safe region and safe period in order to minimize alarm miss-rate. We believe that it is interesting to investigate the effect of alarms with both moving targets and moving subscribers on the success rate, the efficiency and accuracy of alarm processing. Methods for safe region construction and representation in presence of these type of alarms (moving subscriber and moving target) need to be carefully explored.

**Location Privacy for Spatial Alarms:** We have discussed mechanisms for providing location privacy for ad-hoc services (i.e., those that require snapshot queries). Extension of such location anonymization methods and models to handle continuous spatial services is an open challenge.

Furthermore, location-based advertisement or monitoring services supported by the spatial alarms framework require installation of long-standing spatial alarms. Also, spatial alarms may provide additional (and yet sensitive) information associated with an individual's preference. For example, a private alarm on a subscriber's favorite grocery store provides additional information associated with the subscriber, which would not be available in case a snapshot query for neighboring grocery stores was issued.

Although our safe period and safe region techniques, by virtue of limiting the client-server communication, are inherently privacy-aware, the privacy measures used in this dissertation may have limitations when considered in the context of location privacy for spatial alarms. Continuous location tracking may allow an adversary to make certain unexpected inferences about the users location. We argue that there is a need for defining privacy measures which hold in the presence of continuous location updates and for preventing leakage of information due to installation and management of alarms at the centralized server(s).

### 8.2.2 Location Privacy

The availability of a growing amount of personal information and the lack of awareness among users regarding the risks of making their location information available over social networks have resulted in new challenges which are beyond the scope of this dissertation. We believe that such new challenges deserve their own attention and need to be addressed.

**Location Privacy Plug-ins:** It is possible to provide configurable services for emerging applications like presence-based applications, large-scale social networks and location-based event dissemination systems on top of the spatial alarm framework. Location privacy-aware services constitute one such set of services, which can be incorporated as a plug-in. However, naïve users may need some interface that is much more simpler and user friendly. For instance, we can develop a mechanism to allow users to express their privacy needs as simple privacy levels which can be translated to corresponding privacy parameters ($k$-anonymity, $s$-anonymity and $l$-diversity values) dependent on the service request density

in the relevant spatial regions. Corresponding QoS parameters, such as maximum spatial tolerance and maximum temporal tolerance, are also determined based on the user density in a given region and the required privacy levels. In short, the wide-deployment of PrivacyGrid and MobiCloak methods can benefit from plug-ins that can simplify the semantic specification of desired privacy and QoS metrics.

**Further Study on Effects of incorporating Temporal Dimension:** Our work considers two different classes of QoS metrics used in the system: spatial tolerance and temporal tolerance. By temporal tolerance we mean that location cloaking requests can be delayed for a user specified time period. We consider location-based services with user location as the predicate associated with the query. The anonymization algorithms are designed to ensure that the exact location of a user cannot be associated with any particular query which may lead to linking of the spatial query with a particular user. In order to meet this goal, the user location granularity is reduced within the spatial tolerance limits. The methods presented in this thesis involve algorithms for spatial cloaking of the user location with temporal tolerance as a delay criterion acceptable to the user. The deferred cloaking approach allows for introduction of delay in the cloaking process which allows for trade-offs between the relative spatial resolution and the relative temporal resolution.

Spatio-temporal queries would require addition of the temporal dimension as a predicate for query processing which would require our algorithms to treat the temporal dimension in a similar manner as the spatial dimension. For the grid cloaking algorithms in PRIVACYGRID, this would require the two-dimensional algorithms to be extended to three-dimensional algorithms. The current treatment of the temporal dimension assumes the spatial dimension is of primary concern. Furthermore, it is interesting to explore the idea of anonymizing the temporal dimension independent of the spatial dimension.

**Location Privacy in a Broader Context:** Social networking and online communities today allow users to share their location at different time instants through messaging and

presence-based sharing methods. The effects of excessive sharing of user location information for social networking applications, like [13, 18, 17] can be adverse. Furthermore, the lack of awareness among users can lead to harmful effects [19]. We believe that policy-based release of location information can be combined with location anonymization to achieve lower location granularity. It is up to the users to apply the access control mechanisms provided by these applications to ensure that their personal information is shared only with users within their network or the limited set of users with whom the information is intended to be shared. Further research in this area is required for considering the effects of sharing of user location information in a social networking setting.

## *8.3 Concluding Remarks*

There were two ways to go about packaging the dissertation, either as a set of chapters, each dealing with a specific problem related with scalability for location-based services or as a more unified book. For example, the mobility simulator used in each part of this work could possibly be written as a separate chapter. However, each piece of work uses a different version of the mobility simulator, as continuous enhancements were made as the dissertation work progressed. Further, different parameters and data are generated for each piece of work, keeping in mind the evaluation metrics required for measuring the performance of our algorithms. There were arguments in favor of and against each of these approaches. In the end, this dissertation is presented as a set of chapters with a uniform theme as it provides for a more clear and simpler presentation of this dissertation research.

Furthermore, this dissertation research started in 2006 with spatial alarms and location privacy as two major projects, each aiming at solving the scalability issues for location-based and possibly anonymous services. A number of commercial developments have revolutionized in the past four years in the context of location-based services and mobile Internet. Today we see that wireless and mobile computing and location-based services are

becoming a very large and successful industry with tremendous growth potential. Everyone is connected at all times while on the move. Development of powerful handsets and a plethora of location-based services and applications have been the driving force for this dissertation research, developing an architectural infrastructure for developing location-based, event-driven, anonymous mobile systems and services for future computing environments.

# REFERENCES


[1] "Authorities: GPS System Used to Stalk Woman." http://www.usatoday.com/tech/news/2002-12-30-gps-stalker_x.htm, January 2006.

[2] "ABIResearch - Mobile Location-Based Services to Reach $13.3 Billion in 2013." http://www.abiresearch.com/home.jsp, March 2007.

[3] "Anonymous Web Surfing." http://www.anonymizer.com, January 2007.

[4] "Geominder - Unleash the power of location-based reminders." http://ludimate.com/products/geominder/, January 2007.

[5] "Global Mapper Software LLC." http://www.globalmapper.com, February 2007.

[6] "Naggie 2.0: Revolutionize Reminders with Location!." http://www.naggie.com/, January 2007.

[7] "Platform for Privacy Preferences (P3P) Project." http://www.w3.org/P3P/, January 2007.

[8] "U.S. Geological Survey." http://www.usgs.gov, January 2007.

[9] "Scalable Vector Graphics Format." http://www.w3.org/Graphics/SVG, January 2008.

[10] "Spatial Data Transfer Format." http://www.mcmcweb.er.usgs.gov/sdts/, January 2008.

[11] "Android - An Open Handset Alliance Project." http://code.google.com/android/, March 2009.

[12] "Apple Developer Connection - iPhone SDK." http://developer.apple.com/iphone/program/download.html, March 2009.

[13] "Facebook." http://www.facebook.com/, March 2009.

[14] "GPS-Enabled Mobile Devices Market Research Report." http://www.abiresearch.com/products/market_research/GPS_Enabled_Mobile_%Devices, January 2009.

[15] "GPS-Enabled Mobile Devices Market Research Report." http://www.abiresearch.com/products/market_research/GPS_Enabled_Mobile_%Devices, March 2009.

[16] "Palm Pre Phone." http://www.palm.com/us/products/phones/pre/, January 2009.

[17] "foursquare." http://foursquare.com/, January 2010.

[18] "Google Latitude." http://www.google.com/mobile/latitude/, January 2010.

[19] "Please Rob Me." http://pleaserobme.com/, June 2010.

[20] AGGARWAL, C., "On k-Anonymity and the Curse of Dimensionality," in *VLDB*, 2005.

[21] AURENHAMMER, F., "Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[22] BABCOCK, B., BABU, S., DATAR, M., MOTWANI, R., and WIDOM, J., "Models and Issues in Data Stream Systems," in *ACM PODS*, pp. 1–16, 2002.

[23] BABU, S. and WIDOM, J., "Continuous Queries over Data Streams," *ACM SIGMOD Record*, vol. 30, no. 3, pp. 109–120, 2001.

[24] BAMBA, B. and LIU, L., "PRIVACYGRID: Supporting Anonymous Location Queries in Mobile Environments," tech. rep., Georgia Tech., 2007.

[25] BAMBA, B., LIU, L., IYENGAR, A., and YU, P., "Distributed Processing of Spatial Alarms: A Safe Region-based Approach," in *IEEE ICDCS*, pp. 207–214, 2009.

[26] BAMBA, B., LIU, L., IYENGAR, A., and YU, P. S., "Safe Region Techniques for Fast Spatial Alarm Evaluation," tech. rep., Georgia Institute of Technology, 2008.

[27] BAMBA, B., LIU, L., PESTI, P., and WANG, T., "Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid," in *WWW*, 2008.

[28] BAMBA, B., LIU, L., and YU, P. S., "Scalable Processing of Spatial Alarms," tech. rep., Georgia Institute of Technology, 2008.

[29] BAYARDO, R. and AGRAWAL, R., "Data Privacy Through Optimal k-Anonymization," in *ICDE*, 2005.

[30] BAZINETTE, V., COHEN, N., EBLING, M., HUNT, G., LEI, H., PURAKAYASTHA, A., STEWART, G., WONG, L., and YEH, D., "An Intelligent Notification System," *IBM Research Report RC 22089 (99042)*, 2001.

[31] BECKMANN, N., KRIEGEL, H., SCHNEIDER, R., and SEEGER, B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," in *ACM SIGMOD*, pp. 322–331, 1990.

[32] BENTLEY, J. L., "Multidimensional Binary Search Trees Used for Associative Searching," *CACM*, vol. 18, no. 9, pp. 509–517, 1975.

[33] BERCHTOLD, S., KEIM, D., and KRIEGEL, H., "The X-tree: An Index Structure for High-Dimensional Data," in *VLDB*, pp. 28–39, 1996.

[34] BERESFORD, A. and STAJANO, F., "Location Privacy in Pervasive Computing," *Pervasive Computing, IEEE*, 2003.

[35] CAI, Y. and HUA, K. A., "An Adaptive Query Management Technique for Efficient Real-Time Monitoring of Spatial Regions in Mobile Database Systems," in *IEEE IPCCC*, 2002.

[36] CHO, H. and CHUNG, C., "An Efficient and Scalable Approach to CNN Queries in a Road Network," in *VLDB*, 2005.

[37] COMER, D., "The Ubiquitous B-tree," *ACM Computing Surveys*, vol. 11, pp. 121–137, 1979.

[38] COMPUTER SCIENCE AND TELECOMMUNICATIONS BOARD, "IT Roadmap to a Geospatial Future," *The National Academics Press*, 2003.

[39] DAS, A., GEHRKE, J., and RIEDEWALD, M., "Approximate Join Processing over Data Streams," in *ACM SIGMOD*, pp. 40–51, 2003.

[40] DEY, A. and ABOWD, G., "CybreMinder: A Context-Aware System for Supporting Reminders," in *Second International Symposium on Handheld and Ubiquitous Computing*, pp. 172–186, 2000.

[41] DUCKHAM, M. and KULIK, L., "A Formal Model of Obfuscation and Negotiation for Location Privacy," in *Pervasive*, pp. 152–170, 2005.

[42] DURI, S., GRUTESER, M., LIU, X., MOSKOWITZ, P., PEREZ, R., SINGH, M., and TANG, J., "Framework for Security and Privacy in Automotive Telematics," in *Second International Workshop on Mobile Commerce*, 2002.

[43] FALOUTSOS, C., S. T. K. and N., R., "Analysis of object oriented spatial access methods," in *ACM SIGMOD*, 1987.

[44] FLAUTNER, K. and MUDGE, T., "Vertigo: Automatic Performance-Setting for Linux," *Operating Systems Review*, vol. 36, pp. 105–116, December 2002.

[45] FLINN, J. and SATYANARAYANAN, M., "Energy-Aware Adaptation for Mobile Applications," in *SOSP*, pp. 48–63, 1999.

[46] FREUDIGER, J. AND RAYA, M. AND FÉLEGYHÁZI, M. AND PAPADIMITRATOS, P. AND HUBAUX, J.P., "Mix-Zones for Location Privacy in Vehicular Networks," in *Win-ITS*, 2007.

[47] FUJIMOTO, R. M., *Parallel and Distributed Simulation Systems*. Wiley-Interscience, 2000.

[48] GEDIK, B. and LIU, L., "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *IEEE ICDCS*, pp. 620–629, 2005.

[49] GEDIK, B., LIU, L., WU, K., and YU, P., "LIRA: Lightweight, Region-aware Load Shedding in Mobile CQ Systems," in *ICDE*, 2007.

[50] GEDIK, B., WU, K.-L., YU, P. S., and LIU, L., "Motion Adaptive Indexing for Moving Continual Queries over Moving Objects," in *ACM CIKM*, 2004.

[51] GHINITA, G., KALNIS, P., and SKIADOPOULOS, S., "PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems," in *WWW*, 2007.

[52] GOLAB, L. and ÖZSU, M., "Issues in Data Stream Management," *ACM SIGMOD Record*, vol. 32, no. 2, pp. 5–14, 2003.

[53] GRUTESER, M. and GRUNWALD, D., "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *MobiSys*, 2003.

[54] GRUTESER, M. and GRUNWALD, D., "Enhancing Location Privacy in Wireless LAN Through Disposable Interface Identifiers: A Quantitative Analysis," *Mobile Networks and Applications*, 2005.

[55] GUTTMAN, A., "R-trees: A Dynamic Index Structure for Spatial Searching," in *ACM SIGMOD*, 1984.

[56] HONG, J. and LANDAY, J., "An Architecture for Privacy-Sensitive Ubiquitous Computing," in *Mobisys*, pp. 177–189, 2004.

[57] HU, H., XU, J., and LEE, D., "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," in *ACM SIGMOD*, pp. 479–490, 2005.

[58] IWERKS, G., SAMET, H., and SMITH, K., "Continuous K-Nearest Neighbor Queries for Continuously Moving Points with Updates," in *VLDB*, pp. 512–523, 2003.

[59] JAGADISH, H. V., "Spatial Search with Polyhedra," in *IEEE ICDE*, 1990.

[60] JENSEN, C. S., LIN, D., and OOI, B. C., "Query and Update Efficient B+-Tree Based Indexing of Moving Objects," in *VLDB*, 2004.

[61] JONES, K. and LIU, L., "What Where Wi: An Analysis of Millions of Wi-Fi Access Points," in *IEEE Portable*, 2007.

[62] KALASHNIKOV, D. V., PRABHAKAR, S., HAMBRUSCH, S., and AREF, W., "Efficient Evaluation of Continuous Range Queries on Moving Objects," in *DEXA*, 2002.

[63] KAMEL, I. and FALOUTSOS, C., "On Packing R-trees," in *ACM CIKM*, 1993.

[64] KARGER, P. and FRANKEL, Y., "Security and Privacy Threats to ITS," in *World Congress on Intelligent Transport Systems*, pp. 2452–2458, 1995.

[65] KIDO, H., YANAGISAWA, Y., and SATOH, T., "An Anonymous Communication Technique using Dummies for Location-based Services," in *IEEE ICPS*, pp. 88–97, 2005.

[66] KIM, S., KIM, M., PARK, S., JIN, Y., and CHOI, W., "Gate Reminder: A Design Case of a Smart Reminder," in *Conference on Designing Interactive Systems*, pp. 81–90, 2004.

[67] KOLAHDOUZAN, M. and SHAHABI, C., "Voronoi-based k Nearest Neighbor Search for Spatial Network Databases," in *VLDB*, 2004.

[68] KOLLIOS, G., GUNOPULOS, D., and TSOTRAS, V. J., "On Indexing Mobile Objects," in *ACM PODS*, 1999.

[69] KRAMER, J. and J.W., P., "Current Issues in Cell Tower Leases." http://www.lorman.com/bookstore/, 2009.

[70] LANGHEINRICH, M., "Privacy by Design-Principles of Privacy-Aware Ubiquitous Systems," in *UbiComp*, 2001.

[71] LAZARIDIS, I., PORKAEW, K., and MEHROTRA, S., "Dynamic Queries over Mobile Objects," in *EDBT*, 2002.

[72] LEE, M. L., HSU, W., JENSEN, C. S., CUI, B., and TEO, K. L., "Supporting Frequent Updates in R-trees: A Bottom-Up Approach," in *VLDB*, 2003.

[73] LEFEVRE, K., DEWITT, D., and RAMAKRISHNAN, R., "Incognito: Efficient Full-Domain K-Anonymity," in *SIGMOD*, 2005.

[74] LI, N., LI, T., and VENKATASUBRAMANIAN, S., "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *ICDE*, 2007.

[75] LIM, H., LEE, J., LEE, M., WHANG, K., and SONG, I., "Continuous Query Processing in Data Streams using Duality of Data and Queries," in *ACM SIGMOD*, pp. 313–324, 2006.

[76] LIU, L., PU, C., and TANG, W., "WebCQ - Detecting and Delivering Information Changes on the Web," in *CIKM*, pp. 512–519, 2000.

[77] LIU, L., PU, C., and TANG, W., "Continual Queries for Internet Scale Event-Driven Information Delivery," *IEEE TKDE*, pp. 610–628, 1999.

[78] LUDFORD, P., FRANKOWSKI, D., REILY, K., WILMS, K., and TERVEEN, L., "Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications," in *SIGCHI Conference on Human Factors in Computing Systems*, 2006.

[79] MACHANAVAJJHALA, A., GEHRKE, J., KIFER, D., and VENKITASUBRAMANIAM, M., "l-Diversity: Privacy Beyond k-Anonymity," in *ICDE*, 2006.

[80] MARMASSE, N. and SCHMANDT, C., "Location-Aware Information Delivery with ComMotion," in *HUC*, pp. 157–171, 2000.

[81] MEYERSON, A. and WILLIAMS, R., "On the Complexity of Optimal K-Anonymity," in *PODS*, 2004.

[82] MOKBEL, M., CHOW, C., and AREF, W., "The New Casper: Query Processing for Location Services without Compromising Privacy," in *VLDB*, 2006.

[83] MOKBEL, M., XIONG, X., and AREF, W., "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-Temporal Databases," in *ACM SIGMOD*, pp. 623–634, 2004.

[84] MOTWANI, R., WIDOM, J., ARASU, A., BABCOCK, B., BABU, S., DATAR, M., MANKU, G., OLSTON, C., ROSENSTEIN, J., and VARMA, R., "Query Processing, Approximation, and Resource Management in a Data Stream Management System," in *CIDR*, 2003.

[85] MOURATIDIS, K., YIU, M., PAPADIAS, D., and MAMOULIS, N., "Continuous Nearest Neighbor Monitoring in Road Networks," in *VLDB*, 2006.

[86] MUDGE, T., "Power: A First-Class Architectural Design Constraint," *Computer*, vol. 34, no. 4, pp. 52–58, 2001.

[87] MURUGAPPAN, A. and LIU, L., "Energy-Efficient Processing of Spatial Alarms on Mobile Clients," in *SEDE*, 2008.

[88] PAPADIAS, D., TAO, Y., FU, G., and SEEGER, B., "An Optimal and Progressive Algorithm for Skyline Queries," in *SIGMOD*, 2003.

[89] PATEL, J. M., CHEN, Y., and CHAKKA, V. P., "STRIPES: An Efficient Index for Predicted Trajectories," in *ACM SIGMOD*, 2004.

[90] PESTI, P., BAMBA, B., DOO, M., LIU, L., PALANISAMY, B., and WEBER, M., "GTMobiSIM: A Mobile Trace Generator for Road Networks," tech. rep., Georgia Institute of Technology, 2009.

[91] PFITZMANN, A. and KOHNTOPP, M., "Anonymity, Unobservability, and Pseudonymity-A Proposal for Terminology," *Design Issues in Anonymity and Unobservability*, 2000.

[92] PFOSER, D., JENSEN, C. S., and THEODORIDIS, Y., "Novel Approaches in Query Processing for Moving Object Trajectories," in *VLDB*, 2000.

[93] PRABHAKAR, S., XIA, Y., KALASHNIKOV, D., AREF, W., and HAMBRUSCH, S., "Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects," *IEEE Transactions on Computers*.

[94] SALTENIS, S., JENSEN, C. S., LEUTENEGGER, S. T., and LOPEZ, M. A., "Indexing the Positions of Continuously Moving Objects," in *ACM SIGMOD*, 2000.

[95] SAMET, H., *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Reading, Mass, 1990.

[96] SELLIS, T., ROUSSOPOULOS, N., and FALOUTSOS, C., "The $R^+$-tree: A Dynamic Index for Multidimensional Objects," in *VLDB*, 1987.

[97] SOHN, T., LI, K., LEE, G., SMITH, I., SCOTT, J., and GRISWOLD, W., "Place-Its: A Study of Location-Based Reminders on Mobile Phones," in *UbiComp*, 2005.

[98] SONG, Z. and ROUSSOPOULOS, N., "k-Nearest Neighbor Search for Moving Query Point," in *SSTD*, 2001.

[99] SWEENEY, L., "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[100] SWEENEY, L., "k-Anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002.

[101] TAN, K., ENG, P., and OOI, B., "Efficient Progressive Skyline Computation," in *VLDB*, 2001.

[102] TAO, Y. and PAPADIAS, D., "Time-parameterized Queries in Spatio-temporal Databases," *ACM SIGMOD*, pp. 334–345, 2002.

[103] TAO, Y., PAPADIAS, D., and SUN, J., "The $TPR^*$-Tree: An Optimized Spatio-temporal Access Method for Predictive Queries," in *VLDB*, 2003.

[104] TATBUL, N., ÇETINTEMEL, U., ZDONIK, S., CHERNIACK, M., and STONEBRAKER, M., "Load Shedding in a Data Stream Manager," in *VLDB*, pp. 309–320, 2003.

[105] THEODORIDIS, Y., STEFANAKIS, E., and SELLIS, T., "Efficient Cost Models for Spatial Queries Using R-trees," *IEEE TKDE*, vol. 12, no. 1, pp. 19–32, 2000.

[106] VIGLAS, S., NAUGHTON, J., and BURGER, J., "Maximizing the Output Rate of Multi-way Join Queries over Streaming Information Sources," in *VLDB*, pp. 285–296, 2003.

[107] VIREDAZ, M., "The Itsy Pocket Computer," *Technical Note TN-54, WRL, Compaq*, 1998.

[108] WANG, T. AND LIU, L., "Privacy-Aware Mobile Services over Road Networks," in *VLDB*, 2009.

[109] WOLFSON, O., "Moving Objects Information Management: The Database Challenge," in *Next Generation Information Technologies and Systems*, 2002.

[110] WU, K.-L., CHEN, S.-K., and YU, P. S., "Processing Continual Range Queries over Moving Objects Using VCR-Based Query Indexes," in *Mobiquitos*, 2004.

[111] XIAO, X. and TAO, Y., "Personalized Privacy Preservation," in *SIGMOD*, 2006.

[112] XIAO, X. and TAO, Y., "m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets," in *SIGMOD*, 2007.

[113] XIONG, X., MOKBEL, M. F., and AREF, W. G., "SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases," in *IEEE ICDE*, 2005.

[114] XU, J., TANG, X., and LEE, D., "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," *IEEE TKDE*, pp. 474–488, 2003.

[115] YU, X., PU, K., and KOUDAS, N., "Monitoring k-Nearest Neighbor Queries over Moving Objects," in *ICDE*, pp. 631–642, 2005.

[116] ZHANG, J., ZHU, M., PAPADIAS, D., TAO, Y., and LEE, D., "Location-based Spatial Queries," *ACM SIGMOD*, pp. 443–454, 2003.

# VITA

Bhuvan Bamba was born and brought up in New Delhi, the capital city of India. He received a Bachelor of Technology degree from Indian Institute of Technology, Madras, India in 2003. This was followed by a short stint at IBM India Research Lab in New Delhi where the foundation for his research ambition was laid. After two fruitful years at IBM, Bhuvan moved to Atlanta to pursue a Ph.D. in Computer Science at the College of Computing at Georgia Institute of Technology. Bhuvan pursued his dissertation research in the area of Location-based Services and Systems under the guidance of Prof. Ling Liu in the Distributed Data Intensive Systems Lab. He contributed to a multitude of projects in the area of location-based services and applications, privacy and distributed systems. His internships in the industry with IBM and NTT cover areas in mobile computing, data management and information retrieval. His list of honors includes the Dean's Fellowship from Georgia Institute of Technology in 2006 and a best paper award from the ACM Conference on Information and Knowledge Management in 2005. His work in the industry has led to numerous patent filings and a high value patent award from IBM.