

# A Motion-Aware Safe Period-based Framework for Spatial Alarm Processing

Bhuvan Bamba<sup>◇</sup>      Ling Liu<sup>◇</sup>      Philip S. Yu<sup>♣</sup>

<sup>◇</sup> College of Computing, Georgia Institute of Technology

<sup>♣</sup> Department of Computer Science, University of Illinois at Chicago  
{bhuvan, lingliu}@cc.gatech.edu, psyu@cs.uic.edu

## Abstract

Spatial alarms are set on a *spatial location of interest* which the subscribers of the alarm will travel to sometime in the future. 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. In this paper, we propose to use *safe period optimizations* to minimize the number of unnecessary spatial alarm evaluations, increasing the throughput and scalability of the system. We also develop a suite of alarm grouping techniques, based on spatial locality of the alarms and motion behavior of the mobile users, which reduce the safe period computation costs. Subscriber mobility-based optimizations are introduced to further enhance the performance of the system. An evaluation of our approach using a road network simulator shows that the proposed motion-aware safe period-based approach to spatial alarm processing offers significant performance enhancements.

## 1 Introduction

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.

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. A conventional approach would involve 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 evaluation 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.

Spatial alarms can be classified into three categories based on the publish-subscribe scope of the alarm as *private*, *shared* or *public* 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. Mobile users may subscribe to *public* alarms by topic

categories or keywords, such as “*traffic information on highway 85North*”, “*top ranked local restaurants*”, to name a few. Without loss of generality, the rest of the paper 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. Instead of pushing a large number of public alarms to all interested clients, we consider a system which performs alarm processing on the server side.

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.

Many location-aware systems use centralized solutions for a variety of reasons. A lot of the information required by the system can generally be available only at a server and not on all clients. Thus, it is essential for the clients to report their local state (position coordinates) to a central server. In our setting, a client-based solution works perfectly for private alarms. However, in order to enable alarm sharing and to evaluate public alarms it is essential to have a centralized server for alarm monitoring. Further, this work discusses only the spatial aspect of the alarm processing system. However, the predicate conditions for spatial alarms generally involve non-spatial attributes too. This will involve monitoring of large number of attributes at the server being delivered by different data sources. Obviously, this information cannot be delivered to all clients continuously.

In this paper, 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 propose to use 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 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.

The rest of the paper is outlined as follows. Section 2 provides a formal description of the spatial alarm problem and describes the concepts associated with the same. Our spatial alarm processor architecture is also introduced here. After considering the various options for spatial alarm processing and discussing their weaknesses, we describe the safe period computation technique in section 3. Next, we introduce various alarm grouping-based optimization techniques in section 4 followed by the subscriber mobility-based optimizations in section 5. Our experimental evaluation results are detailed in section 6 followed by a brief description of related work in section 7. We conclude the paper in section 8.

## 2 System Overview

In this section, we discuss concepts associated with spatial alarms and then describe our server-based system architecture. In addition, we describe two alternative ways of processing spatial alarms discussing pros and cons of each, followed by an introduction to the concept of safe period and its benefits for alarm evaluation.

## 2.1 Spatial Alarms

A mobile user can define and install many spatial alarms; each alarm is typically subscribed to by one or many 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. Each alarm is associated with an *alarm target* which specifies the location of interest to the user; a region surrounding the alarm target is defined as the *spatial alarm region*. The *alarm trigger* condition requires that subscribers of the alarm be notified as soon as they enter the spatial alarm region.

In addition to the classification based on the publish-subscribe scope, 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, etc. The second class is referred to as the *Static Subscribers Mobile Targets* (SSMT) 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*”.

## 2.2 System Architecture

We assume that mobile users update their positions continuously through periodic location updates or other location estimation techniques based on known speed, elapsed time and motion parameters. The proposed spatial alarm processing system architecture is shown in Figure 1, and it consists of three main components: *alarm installation or removal, alarm evaluation and optimization, and alarm notification and delivery*. The

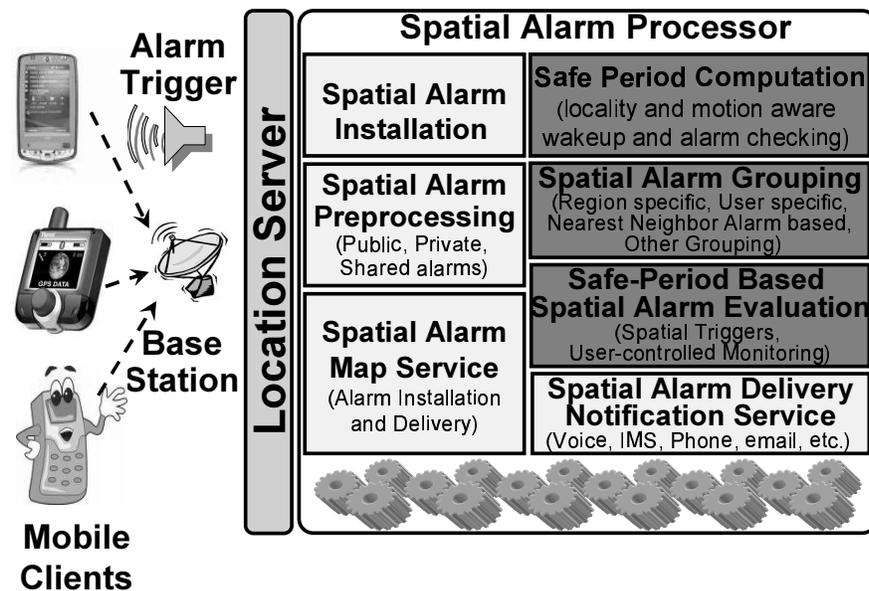


Figure 1: Spatial Alarm Processor Architecture

**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 removal of existing alarms. Only the alarm owner is authorized to delete the active alarms she has installed.

The **alarm evaluation and optimization** 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

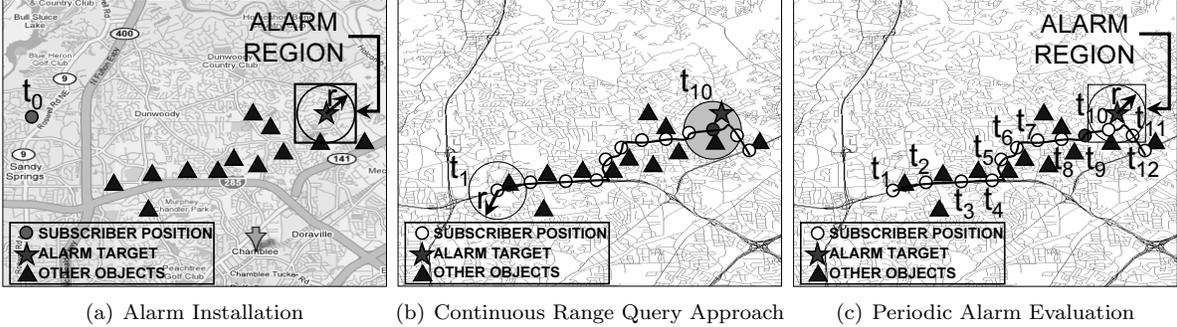


Figure 2: Naive Evaluation Techniques

period computation and alarm grouping are performed in the optimization phase. In the third phase, known as runtime execution, the actual alarm evaluation takes place. 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 a 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. 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, video etc.).

### 2.3 Spatial Alarm Processing

We dedicate this section to discuss the weaknesses of existing spatial query processing techniques [26, 25] 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 discussion. Figure 2(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 radius  $r$  around the alarm target.

Figures 2(b) and 2(c) display the road network for the same region shown in Figure 2(a), extracted from the United States Geological Survey (USGS) [6] database, which we use for experimental evaluation. As shown in Figure 2(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 2(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.

Alternatively, we can use a *periodic alarm evaluation* technique as shown in Figure 2(c). At each time instant from  $t_1$  onwards, the system needs to determine if the current object position lies within the Minimum Bounding Rectangle (MBR) of the spatial alarm region. In case alarm evaluation is performed at an interval of one minute, periodic 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 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 4 and 5.

### 3 Safe Period Computation

*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 an alarm target  $A_j$  ( $1 \leq i \leq N$ ,  $1 \leq j \leq M$ ), 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  $A_j$  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  $R_j$  associated with  $A_j$ , 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  $S_i$  denoted by  $f(V_{S_i})$ , and (ii) the distance  $d(S_i, R_j)$  from the current position of subscriber  $S_i$  to the spatial alarm region  $R_j$ . Thus, the safe period  $sp(S_i, A_j)$  can be computed as follows:

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_{S_i})} \quad (1)$$

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

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_T)} \quad (2)$$

Clearly, the distance measure between the current location of the subscriber and the 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.

#### 3.1 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

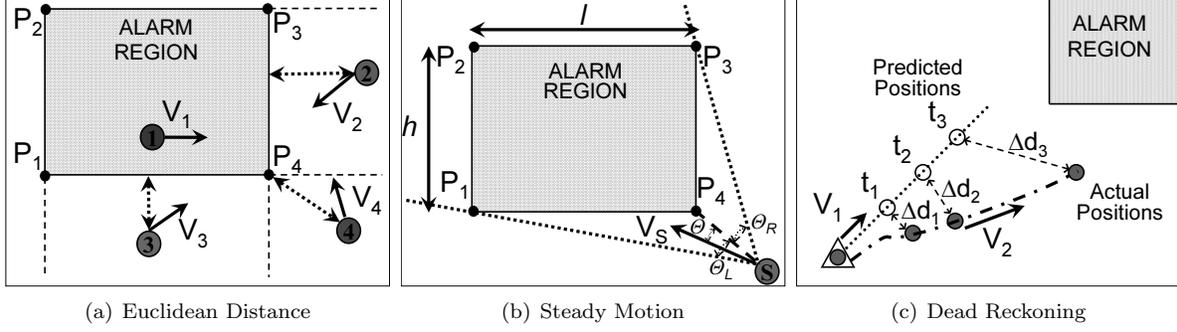


Figure 3: Basic Safe Period Computation

alarm region  $R$ . Though the Euclidean distance measurement is simple, it may at times underestimate the safe period for a given alarm-subscriber pair.

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 3(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 vertices. 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:

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

$d_{m,k}$ ,  $k \in \{1, 2, 3, 4\}$  denotes the Euclidean distance from  $P_m$  to rectangle vertex  $P_k$ . 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 and 2 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 3(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,

$$sp = \frac{\int_{-\theta_L}^{+\theta_R} sm_{dist}(\theta) d\theta}{f(V_S) \int_{-\theta_L}^{+\theta_R} d\theta} = \frac{l + h}{f(V_S)(\theta_R + \theta_L)}, \quad (3)$$

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.

### 3.2 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.

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 case 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 [28] which require the mobile user to report to the server when her velocity increases over a certain threshold, as shown in Figure 3(c). The use of dead reckoning or similar techniques will allow the server to recompute the safe period for all alarms subscribed to by this mobile client upon any significant velocity change.

In Figure 3(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  $(1 - \beta)$ ).

$$V_{expected}^c = \beta * \frac{D(l_c, l_p)}{(t_c - t_p)} + (1 - \beta) * V_{expected}^p \quad (4)$$

$$V_{expected} = \alpha * V_{expected}^c + (1 - \alpha) * V_S \quad (5)$$

where  $V_{expected}^p$ ,  $V_{expected}^c$ ,  $V_{expected}$  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 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

**Algorithm 1:** Safe Period-based Processing

```

1 forall ( $\vec{p}_s(t), s \in S$ ) do
2   if ( $t < sp(s)$ ) then
3     drop( $\vec{p}_s(t)$ ); //client does not transmit update
4   end
5   else
6     process( $\vec{p}_s(t)$ );
7     forall ( $a_j \in A_s, j \in [1..|A_s|]$ ) do
8       |  $ap(s, r_j) = \text{calcApproachPeriod}(s, r_j)$ ;
9     end
10    |  $sp(s) = \min_{1 \leq j \leq |A_s|} (t + ap(s, r_j))$ ;
11  end
12 end

```

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.

Algorithm 1 provides a high level description of safe period-based processing approach. For each location update  $p_s(t)$ , where  $s \in S$  is a subscriber in the spatial alarm processing system, the algorithm works as follows. If the timestamp of the update  $t$  is less than the safe period  $sp(s)$  of the subscriber, the subscriber avoids transmitting the location update (lines 2-4). Otherwise, the location update is sent to the server and processed against the stored spatial alarm information (spatial indexes) in order to determine if any alarms need to be triggered (lines 5-6). Additionally, the system computes a new safe period for the subscriber  $s$  by considering all alarms  $a_j \in A_s$  relevant to the subscriber. The approach period for the current subscriber position to each relevant alarm region  $r_j$  is calculated by considering distance measurements and velocity estimates as described in the previous sections (lines 7-9). The safe period for the subscriber is calculated as the minimum of the approach period for all relevant alarm regions  $\in A_s$  (line 10).

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. 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.

## 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. Alternatively, we apply spatial locality based-grouping to alarms of each individual subscriber. Our experimental study shows that this approach is more effective. The third locality-based alternative is to employ the nearest alarms-based grouping, which is effective but costly when there are frequent alarm additions and removals.

### 4.1 Spatial Locality-based Grouping

Spatial locality-based (SL) grouping considers the set of alarms from all users and groups together the nearby alarms. This approach outperforms basic safe period alarm evaluation if each group has a large number of alarms belonging to the same subscriber. Figure 4(a) displays the alarm regions for a set of installed alarms.

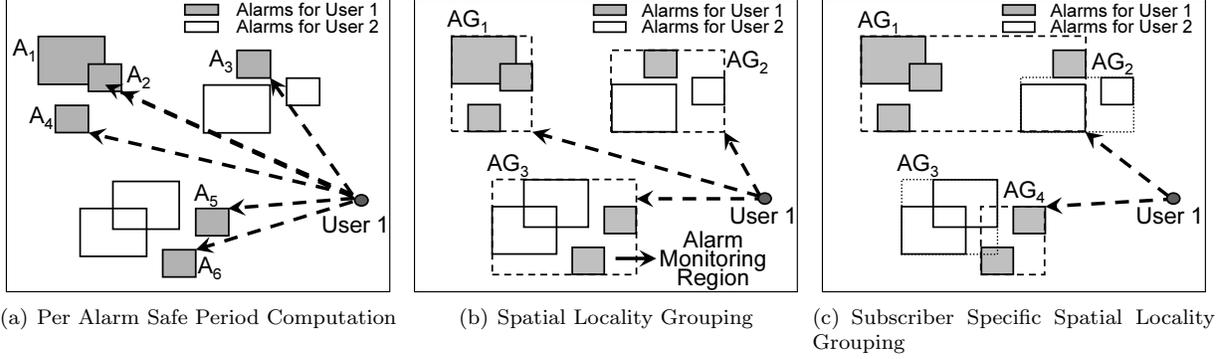


Figure 4: Alarm Locality-based Grouping

The alarms for user 1 are marked by shaded regions. Basic safe period evaluation computes the distance from each of the six alarms  $\{A_i \mid 1 \leq i \leq 6\}$ . In comparison, Figure 4(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 distance for each alarm-subscriber pair, spatial locality-based grouping calculates the distance for each subscriber and alarm group pair. However, on entering a monitoring region the distance 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 computations required by the basic safe period technique, only three 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 computations are dependent on the number of relevant alarms within the user's current alarm monitoring region. Even though this approach reduces the number of 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. This technique proves to be efficient in presence of large number of public alarms as the effect of subscriber specificity is reduced in this situation.

## 4.2 Subscriber Specific Spatial Locality-based Grouping

In contrast to spatial locality-based grouping, subscriber specific spatial locality-based (SSSL) 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. The underlying data structure is a hybrid structure which uses a B-tree for subscriber search at the first level and an R-tree for subscriber specific spatial alarm search at the second level. Figure 4(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 previous approaches. 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.

## 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

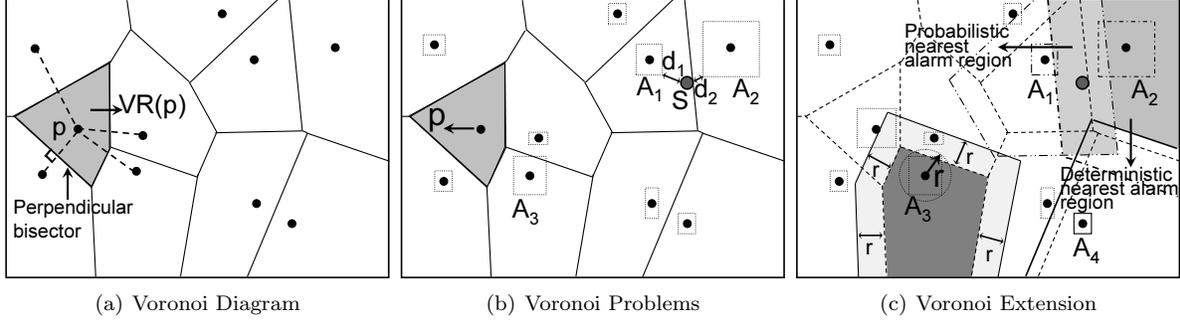


Figure 5: Nearest Alarms-based Grouping

alarm region(s). In order to perform nearest alarms-based grouping, we use an extension of the well known Voronoi diagram geometric structure [7]. 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 *Voronoi region*  $VR(p)$  corresponding to any point  $p \in P$  contains all points  $p_i \in \mathbb{R}^d$  such that,

$$\forall p' \in P, p' \neq p, \text{dist}(p_i, p) \leq \text{dist}(p_i, p') \quad (6)$$

Figure 5(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  $p$ .

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 5(b). But alarm regions may overlap with adjacent Voronoi regions as for alarm  $A_3$  in the figure. Also, 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  $d_2 < d_1$ ,  $A_1$  is incorrectly identified 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 5(c). Extending the Voronoi region boundaries leads to overlaps among neighboring Voronoi regions; subscribers inside overlapping regions (*probabilistic nearest alarm region*) may have more than one possible nearest alarm whereas subscribers inside non-overlapping regions (*deterministic nearest alarm region*) can have only one nearest alarm.

Nearest alarm grouping is efficient for systems that have infrequent addition or removal of alarms and have no hotspots. However, it fails when there is frequent addition/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 this technique.

#### 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 (SL) and the subscriber-specific spatial locality-based (SSSL) approach. As mentioned previously, SL 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 *fanout* 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. The fanout of an index structure implies the maximum number of data entries any node in the index structure may contain. Fill factor of an

index structure is a measure of the fraction of index page that is filled when the index is built. We develop a model which allows us to estimate the appropriate fanout 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  $M$  spatial alarms installed in the system and  $N$  users each having an average of  $m$  relevant alarms. However, note that  $M \neq Nm$  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,

$$N_{bsp} = N_A = M \cdot \left\{ \frac{p}{N} + \frac{s \cdot x}{N} + (1 - p - s) \right\} \quad (7)$$

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 fanout 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,

$$N_{sssl} = \frac{N_A}{b_{br} \cdot ff} + b_{br} \cdot ff \quad (8)$$

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 [15]. 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

$$\frac{d(N_{sssl})}{db_{br}} = -\frac{N_A}{b_{br}^2 \cdot ff} + ff = 0, \quad (9)$$

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

$$N_{sssl} = \frac{N_A}{\frac{\sqrt{N_A}}{ff} \cdot ff} + \frac{\sqrt{N_A}}{ff} \cdot ff = 2 \cdot \sqrt{N_A} = 2 \cdot \sqrt{N_{bsp}} \quad (10)$$

A similar analysis of the SL approach shows that,  $N_{sl} \geq 2 \cdot \sqrt{N_{bsp}}$ . However, as this approach mixes alarms from different users in an alarm group the worst case number of safe period computations can be  $N_{sl} = N_{bsp} + 1$ . 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.

## 5 Subscriber Mobility-based Optimizations

In this section, we introduce subscriber mobility-based optimizations, which further reduce the amount of unnecessary computation. The main idea behind the subscriber mobility-based optimization is to avoid distance computation for alarms that are far away from the current location of the mobile subscriber. We observe that computing the distance for all alarms regardless of how far away they are from the subscriber

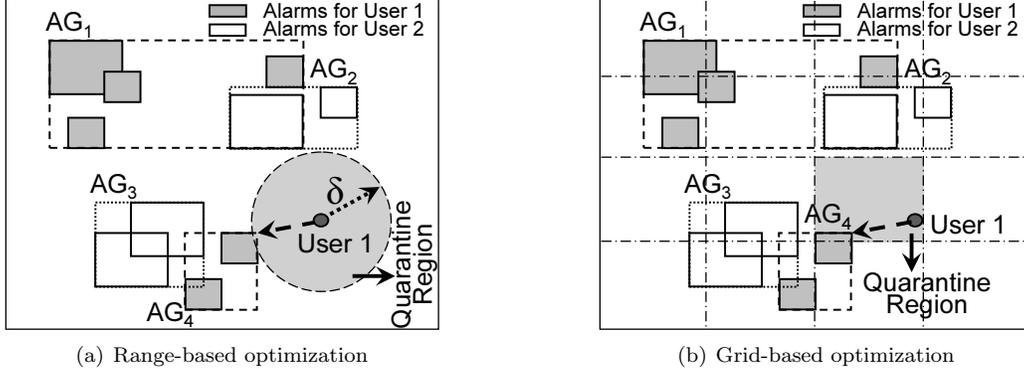


Figure 6: Subscriber Mobility-based Optimizations

position can lead to wasteful computation, since for each subscriber, alarms at remote distances will never be fired before the expiration of the safe periods of nearby alarms.

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, distances are computed only for relevant alarm groups (or alarms) whose alarm group MBRs (or alarm regions) overlap with this moving quarantine region. 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 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 optimization*. Given a mobile subscriber, range-based optimization identifies which of her alarm groups are intersecting with the quarantine region defined by the quarantine radius. We compute the safe period only for those alarm groups whose alarm monitoring regions overlap with the quarantine region (see figure 6(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. 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 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. Recomputation is required only when the subscriber moves out of the current cell and enters another cell.

## 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 100% accuracy. The implementation of each approach was done in Java SDK 1.6 and the experiments were performed on a lenovo T61 laptop with an Intel Centrino Duo 2.00GHz processor and 2GB RAM.

## 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) [6] in Spatial Data Transfer Format (SDTS) [5]. The simulator extracts the road network information for three different road classes – *expressway*, *arterial* and *collector* roads. Traffic volume data from [14] 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 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. 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. Each shared alarm is shared by three to seven users. This simulator setup allows us to test the robustness of our framework under realistic mobility patterns.

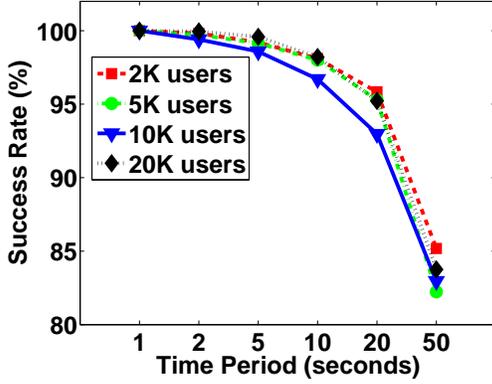
## 6.2 Experimental Results

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. The second set of experiments compares the basic safe period alarm evaluation against the periodic approach. The third set of experiments studies 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 briefly summarize the experimental results below:

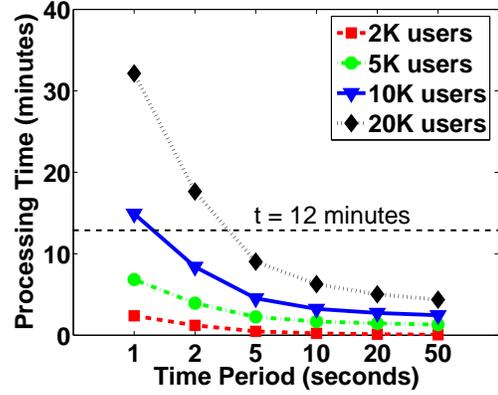
- The periodic alarm evaluation technique has scalability issues as we increase the number of users or installed alarms in the system. It also fails to provide 100% success rate in most scenarios. Results are as displayed in figures 7 and 8.
- The basic safe period approach provides 100% success rate as opposed to the low success rate of the periodic alarm evaluation approach. However, the safe period computation costs are prohibitive in presence of large number of users or large number of alarms. Results are as displayed in figures 9 and 10. This points to the need for the safe period computation optimizations introduced in our work.
- The last set of experiments displays the results for the different safe period optimizations introduced in our work. We show that the Voronoi Grid-based (VG) and the Range-based (RB) subscriber mobility optimizations outperform all other safe period optimization approaches. Results are as displayed in figures 12 and 13.

### 6.2.1 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 7 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 7(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 trace generated with

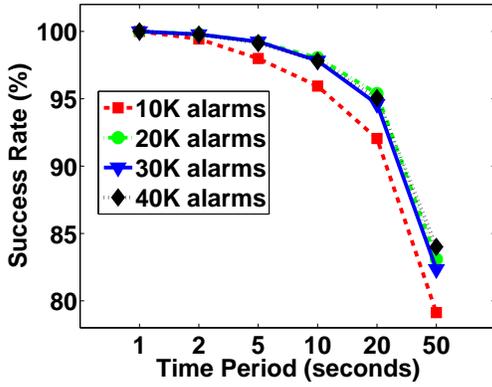


(a) Success Rate

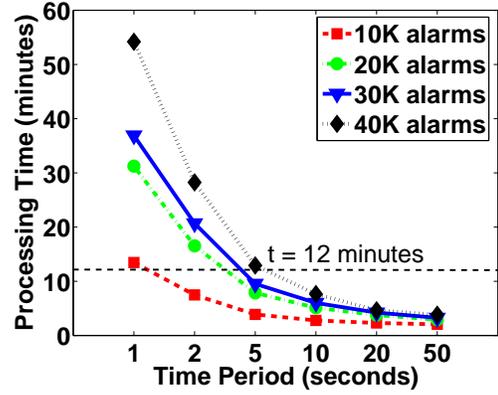


(b) Alarm Processing Time

Figure 7: Scalability with Varying Number of Users



(a) Success Rate



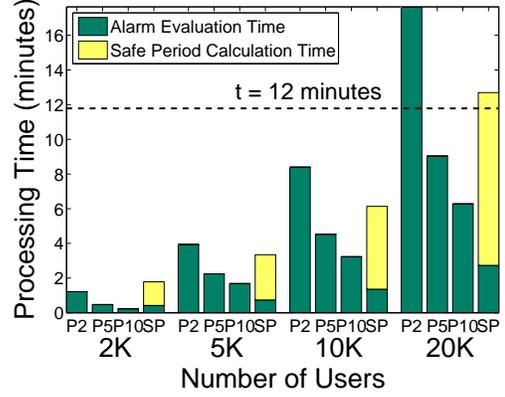
(b) Alarm Processing Time

Figure 8: Scalability with Varying Number of Alarms

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 7(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 7(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 8 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 8(a), again exhibits a similar drop on increasing  $t_p$ . The alarm processing time shown in Figure 8(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.

Number of Users	Basic Safe Period	Periodic 2 seconds	Periodic 5 seconds	Periodic 10 seconds
2K	100	99.8162	99.2039	98.1016
5K	100	99.7144	99.1434	98.0012
10K	100	99.4033	98.5812	97.6984
20K	100	98.9501	98.5800	97.2062

(a) Success Rate (%)

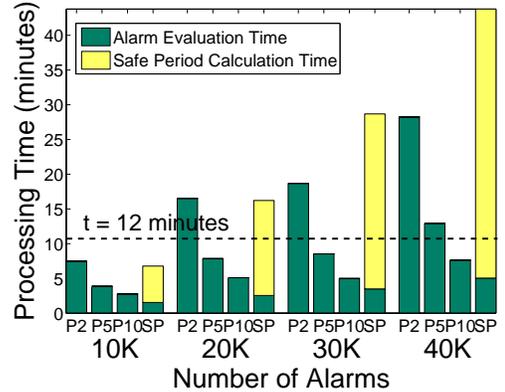


(b) Alarm Processing Time

Figure 9: Basic Safe Period Optimization with Varying Number of Users

Number of Alarms	Basic Safe Period	Periodic 2 seconds	Periodic 5 seconds	Periodic 10 seconds
10K	100	99.4333	99.2852	98.9389
20K	100	99.8060	99.2240	98.04
30K	100	99.7765	99.2394	97.8228
40K	100	99.7656	99.1247	97.8015

(a) Success Rate (%)



(b) Alarm Processing Time

Figure 10: Basic Safe Period Optimization with Varying Number of Alarms

### 6.2.2 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 (P2, P5, P10 and SP in Figures 9(b) and 10(b) respectively). Figure 9 displays the success rate and processing time as we vary the number of users from 2K to 20K. Figure 9(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 9(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 10 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 10(a), again displays similar patterns as with varying number of users. The alarm processing time, as shown in Figure 10(b), displays the inability of our basic safe period approach to scale to large number of

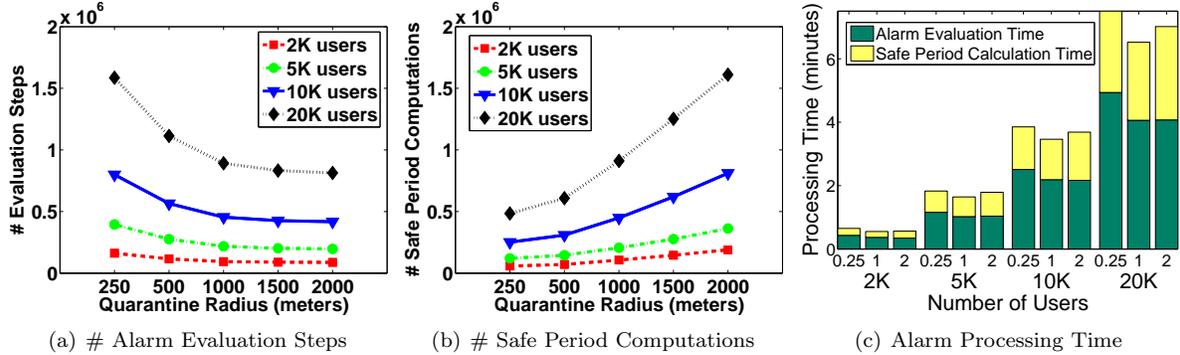


Figure 11: Results with Varying Quarantine Radius Values

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 6.2.4.

### 6.2.3 Internal System Parameters

This set of experiments determines the appropriate parameters for the quarantine radius  $\gamma$  for range-based (RB) subscriber mobility optimization; corresponding results for grid cell size  $\alpha \times \beta$  for Voronoi Grid-based (VG) approach are obtained in a similar manner. Figure 11 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 11(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 number of safe period computations steadily rises as we increase  $\gamma$  (Figure 11(b)). This trend is also as expected because larger  $\gamma$  implies more alarms (or alarm groups) will lie within the quarantine region. A tradeoff between these two factors is required to determine the appropriate value for  $\gamma$ . Figure 11(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  $\gamma \in \{0.25km, 1km, 2km\}$  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  $\gamma = 1$  km. We choose this as the appropriate value for  $\gamma$  for further experimentation.

### 6.2.4 Scalability of Safe Period Evaluation Techniques

We now discuss the performance of the safe period optimization techniques to test the scalability of our framework. Each safe period approach attains 100% accuracy with maximum velocity-based estimates; hence, corresponding accuracy results are excluded from this evaluation. Figure 12 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 4). Hence, we exclude this approach from the results below.

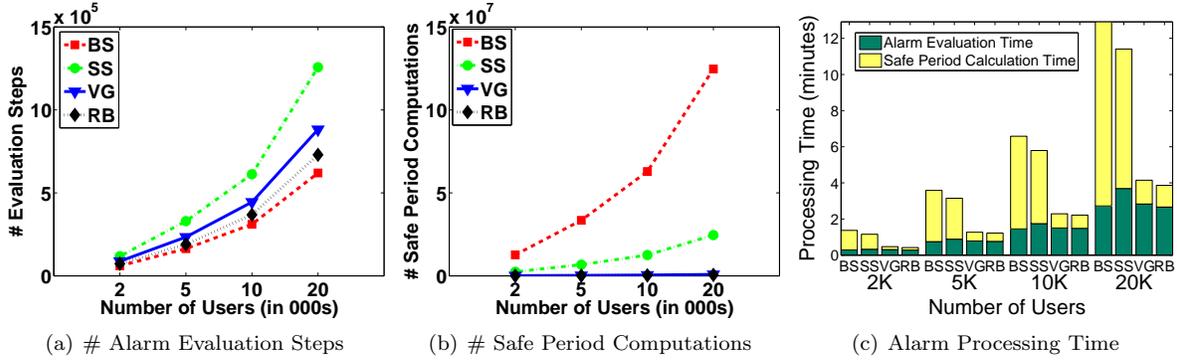


Figure 12: Safe Period Optimizations with Varying Number of Users

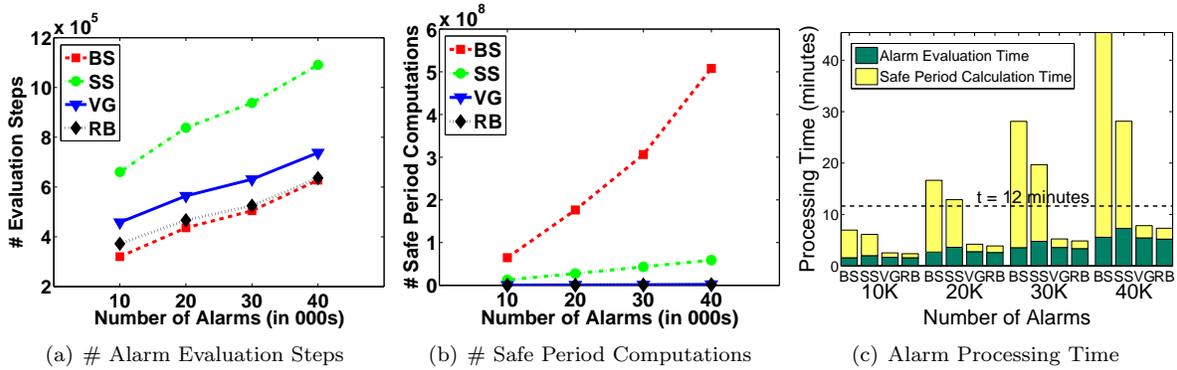


Figure 13: Safe Period Optimizations with Varying Number of Alarms

Figure 12(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 12(b)) which makes this approach overall computationally expensive as can be seen from the total alarm processing times in Figure 12(c). Subscriber-specific spatial locality grouping incurs a large number of alarm evaluation steps as can be seen from Figure 12(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 for 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 the 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 the BS approach as can be seen from Figure 12(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  $1000\text{m} \times 1000\text{m}$ . 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 12(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 13 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 13(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.

## 7 Related Work

An event-based location reminder system has been advocated by many human computer interaction projects [20, 27, 11, 21, 18]. Understandably, the primary focus of the work is from the point of view of the usability of such systems. 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 [19, 10]. 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 [3] and Naggie [4] already exist which provide useful location reminder services using cell tower ID and GPS technology, respectively. Client-based solutions [24] for spatial alarm processing 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 [9, 8] is also essential for extending the technology to clients using cheap location detection devices which may not possess significant computational power. Furthermore numerous works have dealt with the problem of energy conservation in mobile devices [12, 13, 23]. To the best of our knowledge, none has systematically addressed the processing of spatial alarms using an event-based framework. With the development of mobile platforms like Android [1] and iPhone SDK [2], similar applications are available on a multitude of smart phones. However, none of these deal with the system level performance optimization issues which need to be resolved to make such middleware systems and applications feasible.

In spatial and moving object databases research, the focus is on continual spatial queries (kNN, range queries etc.). Spatial alarms are mobile events which require different techniques for scaling and optimization and cannot be handled efficiently using the continuous query paradigm as illustrated in section 2.3. Periodic reevaluation is commonly used for continuous monitoring of moving objects [17, 22, 26, 29]. Some work exists on monitoring continuous queries which applies the concept of *safe region* directly or indirectly [16, 29] for efficient query evaluation. Spatial alarms differ from this work as they do not demand periodic evaluation or reevaluation like continuous queries; instead 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.

Another area which deals with spatial regions is Geographic Information Systems (GIS). In principle, Geographical Information Systems (GIS) work does not deal with mobile objects. GIS combines location and information about the location. GIS provides us the ability to see many 'layers' of information at once and many types of data can be layered together thus providing an ideal framework for analyzing information related to a location.

## 8 Conclusion

We have presented a motion-aware safe period framework and a suite of optimization techniques for scalable processing of spatial alarms. The paper 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.

## Acknowledgement

This work is partially supported by grants from NSF CyberTrust, NSF SGER, NSF CSR, AFOSR, IBM SUR grant and IBM faculty award. The authors would like to thank Bugra Gedik for providing the mobile object simulator.

## References

- [1] Android - An Open Handset Alliance Project. <http://code.google.com/android/>.
- [2] Apple Developer Connection - iPhone SDK. <http://developer.apple.com/iphone/program/download.html>.
- [3] Geominder - Unleash the power of location-based reminders. <http://ludimate.com/products/geominder/>.
- [4] Naggie 2.0: Revolutionize Reminders with Location! <http://www.naggie.com/>.
- [5] Spatial Data Transfer Format. <http://www.mcmcweb.er.usgs.gov/sdts/>.
- [6] U.S. Geological Survey. <http://www.usgs.gov>.
- [7] F. Aurenhammer. Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–405, 1991.
- [8] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu. Distributed Processing of Spatial Alarms: A Safe Reion-based Approach. In *ICDCS*, 2009.
- [9] B. Bamba, L. Liu, P. S. Yu, G. Zhang, and M. Doo. Scalable Processing of Spatial Alarms. In *IEEE HiPC*, 2008.
- [10] V. Bazinette, N. Cohen, M. Ebling, G. Hunt, H. Lei, A. Purakayastha, G. Stewart, L. Wong, and D. Yeh. An Intelligent Notification System. *IBM Research Report RC 22089 (99042)*, 2001.
- [11] A. Dey and G. Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. In *Second International Symposium on Handheld and Ubiquitous Computing*, pages 172–186, 2000.
- [12] K. Flautner and T. Mudge. Vertigo: Automatic Performance-Setting for Linux. *Operating Systems Review*, 36(5S):105–116, December 2002.

- [13] J. Flinn and M. Satyanarayanan. Energy-Aware Adaptation for Mobile Applications. In *SOSP*, pages 48–63, 1999.
- [14] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *MobiSys*, 2003.
- [15] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *ACM SIGMOD*, 1984.
- [16] H. Hu, J. Xu, and D. Lee. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In *ACM SIGMOD*, 2005.
- [17] C. Jensen, D. Lin, and B. Ooi. Query and Update Efficient B+-tree based Indexing of Moving Objects. In *VLDB*, pages 768–779, 2004.
- [18] S. Kim, M. Kim, S. Park, Y. Jin, and W. Choi. Gate Reminder: A Design Case of a Smart Reminder. In *Conference on Designing Interactive Systems*, pages 81–90, 2004.
- [19] L. Liu, C. Pu, and W. Tang. WebCQ - Detecting and Delivering Information Changes on the Web. In *CIKM*, pages 512–519, 2000.
- [20] P. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen. Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 889–898, 2006.
- [21] N. Marmasse and C. Schmandt. Location-Aware Information Delivery with ComMotion. In *HUC*, pages 157–171, 2000.
- [22] M. Mokbel, X. Xiong, and W. Aref. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases. In *ACM SIGMOD*, pages 623–634, 2004.
- [23] T. Mudge. Power: A First-Class Architectural Design Constraint. *Computer*, 34(4):52–58, 2001.
- [24] A. Murugappan and L. Liu. Energy-Efficient Processing of Spatial Alarms on Mobile Clients. In *SEDE*, 2008.
- [25] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *VLDB*, pages 802–813, 2003.
- [26] S. Prabhakar, Y. Xia, D. Kalashnikov, W. Aref, and S. Hambrusch. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. *IEEE Transactions on Computers*, 51(10):1124–1140, 2002.
- [27] T. Sohn, K. Li, G. Lee, I. Smith, J. Scott, and W. Griswold. Place-Its: A Study of Location-Based Reminders on Mobile Phones. In *UbiComp*, 2005.
- [28] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving Objects Databases: Issues and Solutions. In *SSDBM*, pages 111–122, 1998.
- [29] X. Yu, K. Pu, and N. Koudas. Monitoring k-Nearest Neighbor Queries over Moving Objects. In *ICDE*, pages 631–642, 2005.