Scalable Processing of Spatial Alarms

Bhuvan Bamba
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
bhuvan@cc.gatech.edu

Ling Liu
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
lingliu@cc.gatech.edu

Philip S. Yu
Department of Computer Science
University of Illinois at Chicago
Chicago, IL, USA
psyu@cs.uic.edu

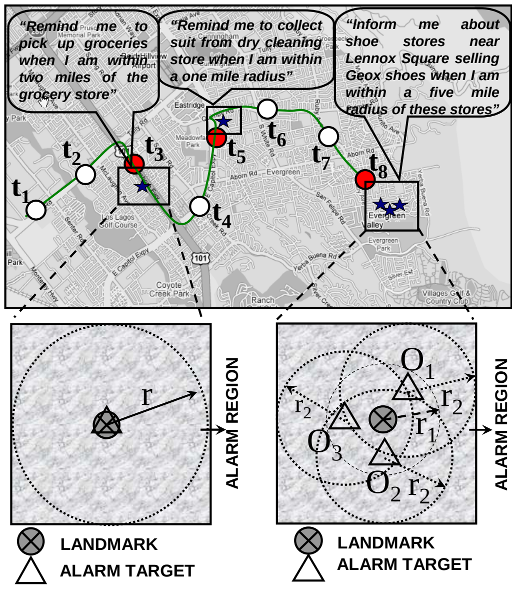
ABSTRACT

Spatial alarms extend the idea of time-based alarms to the spatial dimension. Just as time-based alarms are set to remind us of the arrival of a *future reference time point*, spatial alarms are set on a *spatial location of interest* which the subscribers of the alarm will travel to sometime in the future. 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. In this paper we present a motion-aware framework, facilitated by two systematic methods, for scalable processing of spatial alarms. First, 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. 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 the alarm processing server while maintaining high accuracy of spatial alarms, especially compared to the conventional periodic alarm evaluation approach.

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

(a) Spatial Alarms Example


"Remind me to pick up groceries when I am within two miles of the grocery store"
"Remind me to collect suit from dry cleaning store when I am within a one mile radius"
"Inform me about shoe stores near Lennox Square selling Geox shoes when I am within a five mile radius of these stores"
t_1
t_2
t_3
t_4
t_5
t_6
t_7
t_8
r
ALARM REGION
O_1
O_2
O_3
r_1
r_2
ALARM REGION
LANDMARK
ALARM TARGET
LANDMARK
ALARM TARGET

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

Figure 1: Spatial Alarms

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 1(a) shows three spatial alarms installed on the region around the grocery store at the corner of Clairmont and Baircliff 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. 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 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 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.

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.

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 1(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 1(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 1(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 1(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: 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.

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. 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 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.

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) 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"*.

2.3 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 2, 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.

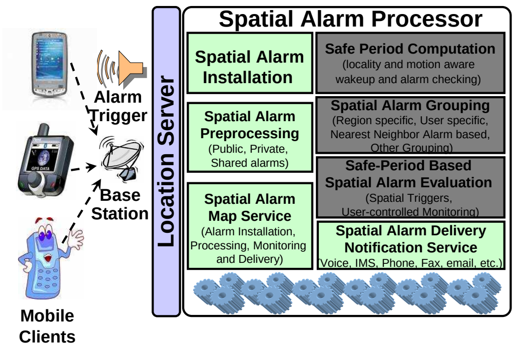

Spatial Alarm Processor
Location Server
Alarm Trigger
Base Station
Mobile Clients
Spatial Alarm Installation
Spatial Alarm Preprocessing
(Public, Private, Shared alarms)
Spatial Alarm Map Service
(Alarm Installation, Processing, Monitoring and Delivery)
Safe Period Computation
(locality and motion aware wakeup and alarm checking)
Spatial Alarm Grouping
(Region specific, User specific, Nearest Neighbor Alarm based, Other Grouping)
Safe-Period Based Spatial Alarm Evaluation
(Spatial Triggers, User-controlled Monitoring)
Spatial Alarm Delivery Notification Service
(Voice, IMS, Phone, Fax, email, etc.)
GPS DATA

Figure 2: Spatial Alarm Processor Architecture

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.).

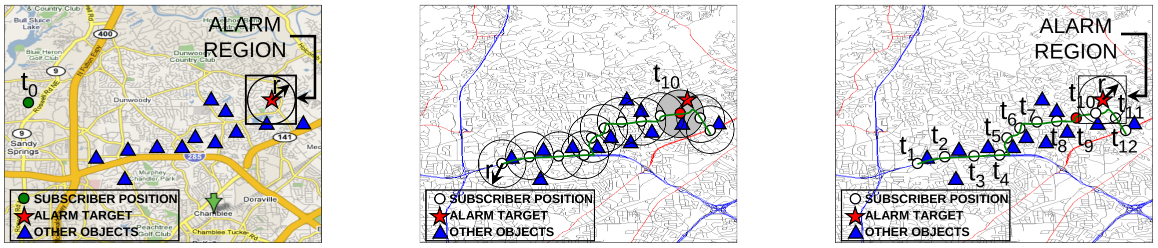
The **spatial alarm evaluation** component works in three phases. First, it accepts spatial alarms as input and indexes them using the underlying spatial indexing structures during the alarm preprocessing phase. Next, the optimization phase applies alarm optimization techniques to produce a near-optimal 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 paper we focus on the design of optimization techniques for spatial alarm evaluation.

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 3(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 3(b) and 3(c) display the road network for the same region shown in Figure 3(a), extracted from the USGS [6] database, which we use for experimental evaluation. As shown in Figure 3(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



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

Figure 3: Naive Evaluation Techniques

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 3(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 3(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 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 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:

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

Similarly, for Static Subscribers Mobile Targets alarm, the safe period $sp(S_i, A_j)$ is computed by taking into account (i) the distance 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:

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

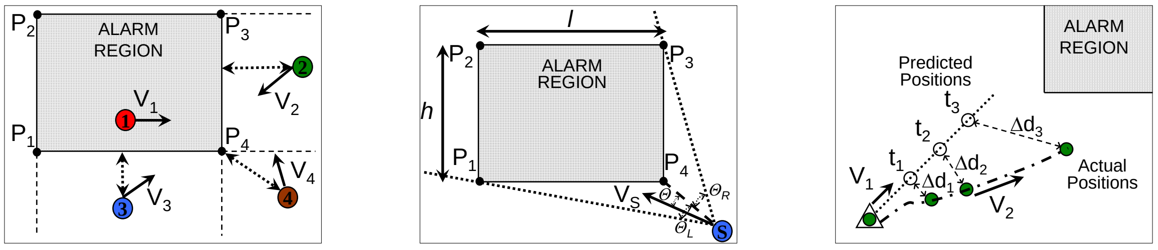
For spatial alarms of class Mobile Subscribers Mobile Targets, the motion characteristics of both subscriber and alarm target need to be considered for the computation of the safe period $sp(S_i, A_j)$, in addition to the distance between the current location of the mobile subscriber and the moving alarm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

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_{S_i}, V_T)} \qquad (3)$$

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.

3.1 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,



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

Figure 4: Basic Safe Period Computation

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 4(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:

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

$d_{m,1}, d_{m,2}, d_{m,3}, d_{m,4}$ 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 4(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 θ 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 θ values ranging from $-\theta_L$ to $+\theta_R$, the steady motion-based safe period over the alarm trigger angular range can be calculated as,

$$sp = \frac{\int_{-\theta_L}^{+\theta_R} sm_{dist}(\theta) d\theta}{f(V_S) \int_{-\theta_L}^{+\theta_R} d\theta} = \frac{l + h}{f(V_S)(\theta_R + \theta_L)}, \qquad (4)$$

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.

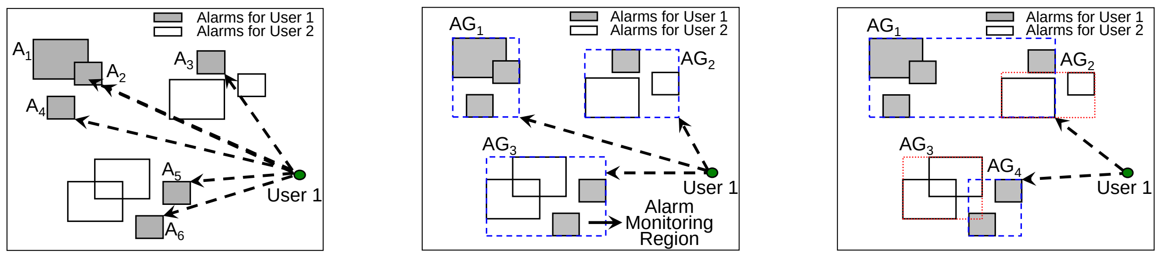
3.2 Velocity Measurements

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 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 4(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 4(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 δ), 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



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

Figure 5: Alarm Locality-based Grouping

future expected travel speed of a mobile client is computed as the sum of the current expected speed weighted by a factor α and the maximum speed weighted by a factor $(1-\alpha)$. Lower α 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 β) and previous expected speed calculation (weighted by $(1-\beta)$).

$$V^{c}_{expected} = \beta * \frac{D(l_c, l_p)}{(t_c - t_p)} + (1-\beta) * V^{p}_{expected} \quad (5)$$

$$V_{expected} = \alpha * V^{c}_{expected} + (1-\alpha) * V_S \quad (6)$$

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 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_δ, 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_δ, 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.

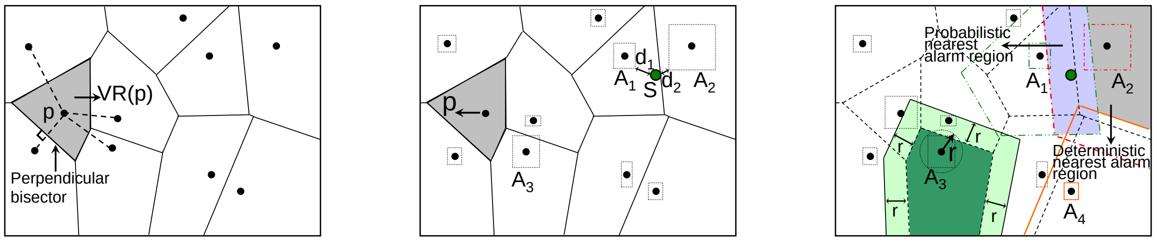
4. ALARM GROUPING TECHNIQUES

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 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 5.

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 5(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 5(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


VR(p)
p
Perpendicular bisector
p
d1
A1
S
d2
A2
A3
Probabilistic nearest alarm region
A1
A2
r
r
A3
r
r
r
Deterministic nearest alarm region
A4

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

Figure 6: Nearest Alarms-based Grouping

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.

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 5(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.

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 [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 ϵ 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,

$$\forall p' \epsilon P, p' \neq p, dist(p_i, p) \leq dist(p_i, p') \qquad (7)$$

Figure 6(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 ϵ 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 6(b). However, this Voronoi structure exhibits two problems. Consider alarm A_3 in Figure 6(b). The alarm region overlaps with 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 6(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.*

THEOREM 4.1. *Objects inside probabilistic nearest alarm region have more than one possible nearest alarm, objects inside deterministic nearest alarm region have a single possible nearest alarm.*

PROOF. Proof skipped due to space limitations. □

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.

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 *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. 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 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,

$$N_{bsp} = N_A = N \cdot \left\{ \frac{p}{m} + \frac{s \cdot x}{m} + (1 - p - s) \right\} \quad (8)$$

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,

$$N_{sssl} = \frac{N_A}{b_{br} \cdot ff} + b_{br} \cdot ff \quad (9)$$

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 [13]. 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

$$\frac{d\,(N_{sssl})}{db_{br}} = -\frac{N_A}{b_{br}^2 \cdot ff} + \cdot ff = 0, \quad (10)$$

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

$$N_{sssl} = \frac{N_A}{\frac{\sqrt{N_A}}{ff} \cdot ff} + \frac{\sqrt{N_A}}{ff} \cdot ff = 2 \cdot \sqrt{N_A} = 2 \cdot \sqrt{N_{bsp}} \quad (11)$$

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.

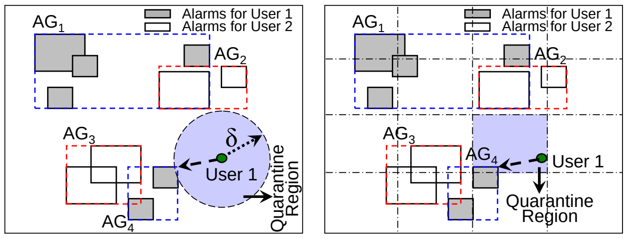
5. SUBSCRIBER MOBILITY-BASED OPTIMIZATIONS

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 γ 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 7 shows examples for these two methods.

For range-based mobility optimization, we currently use a system-supplied γ 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 7(a)). The retrieval of relevant alarms whose alarm regions intersect



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

Figure 7: Subscriber Mobility-based Optimizations

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 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.

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.

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]). 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 [4]) format using the GlobalMapper tool [2]. The simulator extracts the road network information for three different road classes – *expressway*, *arterial* and *collector* roads. Traffic volume data from [12] 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. Table 1 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.

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

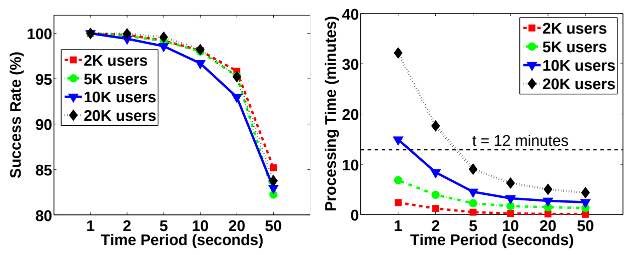
Table 1: Motion Parameters

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. We show that the periodic approach does not scale.The second set of experiments compares the 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.

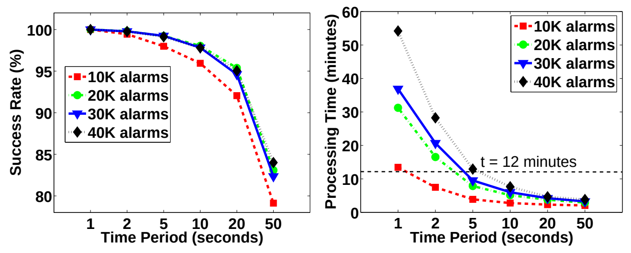
6.2.1 Scalability Problems of Periodic Alarm Evaluation Technique

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 8 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 8(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 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 8(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 8(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 9 shows the results for a set of 10K users as we vary the number of alarms from 10K to 40K. The suc-



(a) Success Rate (b) Alarm Processing Time

Figure 8: Scalability with Varying Number of Users



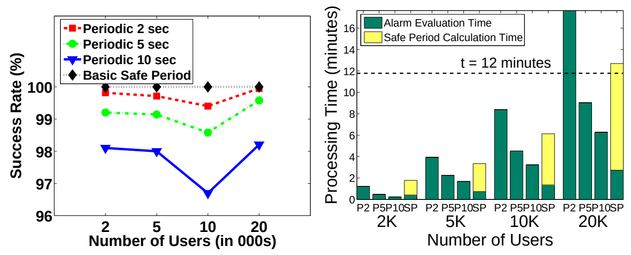
(a) Success Rate (b) Alarm Processing Time

Figure 9: Scalability with Varying Number of Alarms

cess rate, as shown in Figure 9(a), again exhibits a similar drop on increasing t_p. The alarm processing time shown in Figure 9(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.

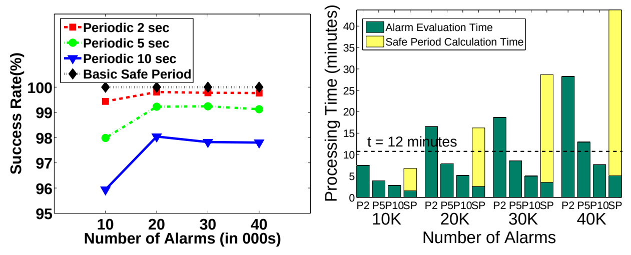
6.2.2 *Performance Comparison with Basic Safe Period Approach*

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 10(b) and 11(b) respectively). Figure 10 displays the success rate and processing time as we vary the number of users from 2K to 20K. Figure 10(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 10(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 11 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 11(a), again displays similar patterns as with varying number of users. The alarm processing time, as shown in Figure 11(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



(a) Success Rate (b) Alarm Processing Time

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



(a) Success Rate (b) Alarm Processing Time

Figure 11: Basic Safe Period Optimization with Varying Number of Alarms

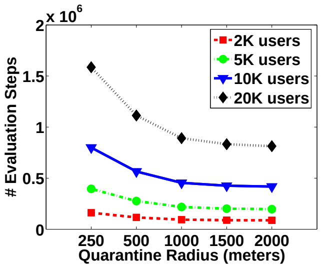
to overcome this problem as displayed by the experimental evaluation in Section 6.2.4.

6.2.3 *Internal System Parameters*

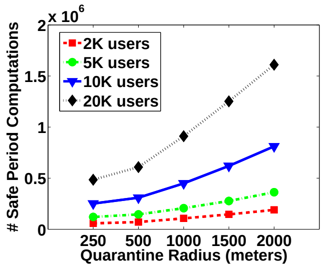
This set of experiments determines the appropriate parameters for the quarantine radius γ for range-based subscriber mobility optimization; results for VGB grid cell size $\alpha \times \beta$ are omitted due to space constraints. Figure 12 displays the results obtained for the number of alarm evaluation steps, number of safe period computations and overall processing time with varying values for γ. 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 12(a) the number of alarm evaluation steps steadily decreases as we increase γ. Smaller γ values will calculate lower safe periods in absence of any alarms (or alarm groups) within the quarantine region. Hence, for lower values of γ 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 γ (Figure 12(b)). This trend is also as expected because larger γ 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 γ. Figure 12(c) displays the overall processing time for different γ 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 γ 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 γ=1000m. We choose this as the appropriate value for γ for further experimentation.

6.2.4 *Scalability of Safe Period Evaluation Techniques*

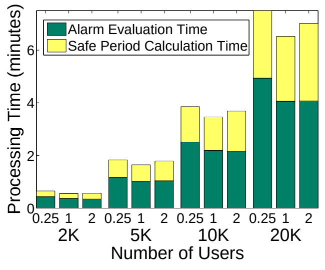
We now discuss the performance of the safe period optimization techniques to test the scalability of our framework. Figure 13 shows the number of alarm evaluation steps,



(a) # Alarm Evaluation Steps

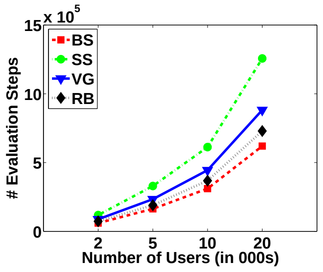


(b) # Safe Period Computations

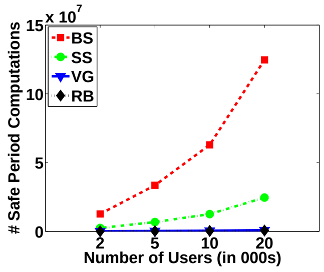


(c) Alarm Processing Time

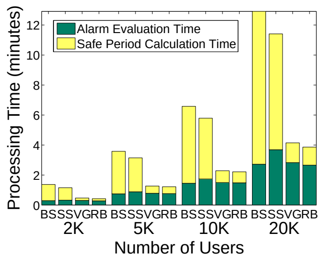
Figure 12: Results with Varying Quarantine Radius Values



(a) # Alarm Evaluation Steps

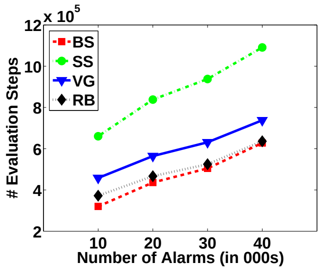


(b) # Safe Period Computations

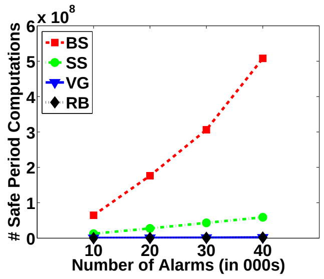


(c) Alarm Processing Time

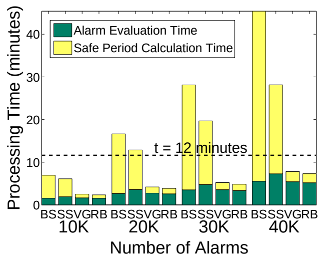
Figure 13: Safe Period Optimizations with Varying Number of Users



(a) # Alarm Evaluation Steps



(b) # Safe Period Computations



(c) Alarm Processing Time

Figure 14: Safe Period Optimizations with Varying Number of Alarms

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 4.1). Hence, we exclude this approach from the results below.

Figure 13(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 13(b)) which makes this approach overall computationally expensive as can be seen from the total alarm processing times in Figure 13(c). Subscriber-specific spatial locality grouping incurs a large number of alarm evaluation steps as can be seen from Figure 13(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 the BS approach as can be seen from Figure 13(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 γ 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 13(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 14 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 14(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 [16, 19, 9, 17, 14]. 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 [15, 8]. 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 [1] and Naggie [3] 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 [10, 11, 18]. In this work, we propose a scalable and efficient centralized architecture for processing spatial alarms to resolve the above issues.

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.

9. REFERENCES

[1] Geominder - Unleash the power of location-based reminders. `http://ludimate.com/products/geominder/`.

[2] Global Mapper Software LLC. `http://www.globalmapper.com`.

[3] Naggie 2.0: Revolutionize Reminders with Location! `http://www.naggie.com/`.

[4] Scalable Vector Graphics Format. `http://www.w3.org/Graphics/SVG`.

[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] 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.

[9] 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.

[10] K. Flautner and T. Mudge. Vertigo: Automatic Performance-Setting for Linux. *Operating Systems Review*, 36(5S):105–116, December 2002.

[11] J. Flinn and M. Satyanarayanan. Energy-Aware Adaptation for Mobile Applications. In *SOSP*, pages 48–63, 1999.

[12] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *MobiSys*, 2003.

[13] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *ACM SIGMOD*, 1984.

[14] 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.

[15] L. Liu, C. Pu, and W. Tang. WebCQ - Detecting and Delivering Information Changes on the Web. In *CIKM*, pages 512–519, 2000.

[16] 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.

[17] N. Marmasse and C. Schmandt. Location-Aware Information Delivery with ComMotion. In *HUC*, pages 157–171, 2000.

[18] T. Mudge. Power: A First-Class Architectural Design Constraint. *Computer*, 34(4):52–58, 2001.

[19] 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.