Distributed Processing of Spatial Alarms: A Safe Region-based Approach

Bhuvan Bamba*, Ling Liu*, Arun Iyengar†and Philip S. Yu‡
*College of Computing, Georgia Institute of Technology
†IBM T.J. Watson Research Center
‡Department of Computer Science, University of Illinois at Chicago
{bhuvan,lingliu}@cc.gatech.edu, aruni@us.ibm.com, psyu@cs.uic.edu

Abstract

Spatial alarms are considered as one of the basic capabilities in future mobile computing systems for enabling personalization of location-based services. In this paper, we propose a distributed architecture and a suite of safe region techniques for scalable processing of spatial alarms. We show that safe region-based processing enables resource optimal distribution of partial alarm processing tasks from the server to the mobile clients. We propose three different safe region computation algorithms to explore the impact of size and shape of the safe region on network bandwidth, server load and client energy consumption. Concretely, we show that the maximum weighted perimeter rectangular safe region approach outperforms previous techniques in terms of performance and accuracy. We further explore finer granularity safe regions by introducing grid-based and pyramid-based representation of rectilinear polygonal shapes using bitmap encoding. Our experimental evaluation shows that the distributed safe region-based architecture outperforms the two most popular server-centric approaches, periodic and safe period-based, for spatial alarm processing.

1. Introduction

Location is becoming an essential communication capability for people to get connected and informed in the wireless and mobile Internet era. Pervasive use of wireless devices, such as smart phones, PDAs and continued price reduction of hardware capable of location sensing and positioning (e.g., GPS, Bluetooth, WiFi) has led to an increasing number of wireless and mobile devices equipped with both positioning capability and communication as well as computation capacity. According to [1], the market for GPS devices is growing at around 20% annually with 90% of the devices being portable navigation devices. By end of 2012, mobile phones equipped with GPS are expected to have around 78% of this market. This makes distributed computing at the edge of the wireless and mobile Internet an attractive alternative for performance optimization and energy saving.

Spatial Alarms. Most of us use time-based alarms to remind us of the arrival of a *future reference time point*; spatial alarms extend the concept of time-based alarms by reminding us of the arrival of a *future spatial location of interest*. The service request *"alert me when I am within two miles of the dry clean store near my house"* is a typical example of spatial alarms. A spatial alarm is defined by three elements: a future location reference known as the alarm target, an owner who is the publisher of the alarm and the list of subscribers of the alarm. We categorize spatial alarms based on two criteria: the publish-subscribe scope of the alarms and the motion characteristics of alarm targets and alarm subscribers. According to the publish-subscribe scope of spatial alarms, we consider three categories of alarms: *private*, *shared* and *public*. *Private* alarms are installed and used exclusively by the publisher. *Shared* alarms are installed by the publisher with a list of authorized subscribers and the publisher is typically one of the subscribers. *Public* alarms are usually installed with the purpose of sharing them with all mobile users who are entering the spatial regions of the alarms. Mobile users may subscribe to public alarms by topic categories or keywords, such as *"traffic information on highway 85 North"* or *"Zagat survey of top-ranked local restaurants"*. Public alarms can be useful means of informing subscribers about hazardous road situations or heavy road congestion. Without loss of generality, the rest of the paper assumes that public alarms are subscribed to by all mobile users. According to the motion characteristics of the alarm target and alarm subscriber, we categorize spatial alarms into three classes: (1) moving subscriber with static target, (2) static subscriber with moving target, and (3) moving subscriber with moving target.

Challenges of Spatial Alarm Processing. 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 processing of spatial alarms. Existing research on spatial alarm processing has been focused on either client-centric [2] or server-centric [3] architectures. A client-centric architecture is limited to supporting only private alarms on static target with moving subscriber or on moving target with static subscriber. This is primarily because other types of spatial alarms require continuous position updates from other mobile clients, which is typically obtained through server-based coordination. In contrast, the server-centric architecture can support all types of spatial alarms and perform alarm processing optimizations. However, with increasing number of users and installed spatial alarms in the system, the alarm processing

server may become a bottleneck.

Paper Scope and Contributions. In this paper, we argue that client-centric and server-centric architectures are not optimal in terms of developing a general framework for scaling spatial alarms-enabled location services. We propose a safe region-based distributed architecture for scalable processing of spatial alarms. We show that our distributed architecture, powered by safe region techniques, can significantly aid scalability, by reducing the amount of unnecessary alarm evaluations required in the server-centric architecture, while maintaining high accuracy. Our approach offers three unique features. First, we introduce the concept of safe region in the context of spatial alarm evaluation. Second, we present the design of a distributed alarm processing partitioning scheme for scaling spatial alarm processing. Our approach optimizes conventional server-centric alarm processing by advocating mobility and locality aware distribution of alarm processing through the safe region-based evaluation framework. More concretely, we compute a safe region for each mobile subscriber or moving alarm target at the server. We further utilize our distributed partitioning scheme to encourage controlled participation of mobile subscribers in safe region monitoring. By distributing the processing of spatial alarms in a controlled fashion between the server and mobile clients, our safe region-based distributed architecture significantly reduces the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system. Last but not the least, we develop a suite of safe region computation techniques to analyze the impact of the size and shape of safe region on the client-server communication cost, server load and client energy consumption. Concretely, we describe three safe region computation techniques: (i) *Maximum Weighted Perimeter Rectangular Safe Region*, (ii) *Grid Bitmap Encoded Safe Region*, and (iii) *Pyramid Bitmap Encoded Safe Region*. These alternative methods for safe region computation provide flexible support for mobile clients with heterogeneous capabilities in terms of CPU, network bandwidth and energy capacity.

In order to validate the effectiveness of our distributed architecture and the safe region-based techniques for scaling spatial alarm processing, we perform experimental comparison of our approach with two popular spatial alarm evaluation approaches: periodic evaluation and safe period-based alarm evaluation. Periodic evaluation can be performed by checking whether a mobile subscriber is entering the spatial region of any of its relevant alarms. High frequency is essential to ensure that none of the alarms are missed. Though periodic evaluation is simple, it can be extremely inefficient due to frequent alarm evaluation and the high rate of irrelevant evaluations. The safe period-based approach [3] allows the system to overcome the deficiencies of periodic evaluation by adaptively computing a safe period for each mobile subscriber; no alarm processing needs to be performed for the mobile user before its safe period expires. However, safe period computation heavily relies on future motion estimation of the mobile user. Our experimental evaluation shows that distributed safe region-based processing outperforms both periodic and safe period-based approaches, while ensuring 100% accuracy.

2. System Overview

In this section, we describe the concept of safe region, outline our distributed safe region-based partitioning scheme and introduce a grid-based model used to restrict the safe region computation costs. Concrete safe region computation techniques are described in subsequent sections.

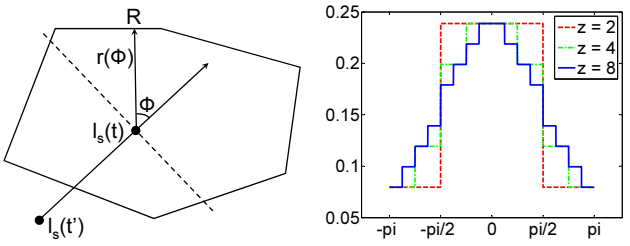
2.1. Safe Region

The use of safe region for distributed processing of spatial alarms is based on a simple observation: *regardless of the total number of relevant spatial alarms for a user, only alarms that are set on objects near her current position have a probability of being triggered*. We show that the safe region approach to spatial alarm processing enables *controlled distribution of alarm processing load between the server and a selection of mobile users*. Such distributed processing can significantly enhance server scalability with nominal resource consumption at the client end.

We can formally characterize the concept of safe region for a given mobile user s, denoted by Ψ_s, as follows: (i) As long as the user's position lies within its safe region, the probability of the user s entering any of its relevant spatial alarm regions is zero. (ii) If the user position lies inside one or more relevant spatial alarm regions, the intersection of the spatial alarm regions forms the safe region for the user s. In this case, the probability of any alarms other than those associated with this safe region being triggered is zero. In summary, as long as a mobile user s resides within its computed safe region, no spatial alarm evaluation is necessary.

The main idea underlying our distributed architecture design is twofold. First, we want to use the concept of safe region to reduce the amount of unnecessary alarm monitoring and alarm checks as the mobile subscribers travel on the road. Second, we promote the distribution of safe region-based alarm monitoring by the mobile clients; each mobile client determines by itself whether it has moved outside its safe region without requiring global knowledge of relevant spatial alarms and the positions of objects of interest. An immediate advantage of our safe region-based distributed architecture is significant savings in terms of server load and communication bandwidth. The main challenge in the design of our distributed architecture is the development of safe region computation techniques that can provide a careful trade-off between server load and client energy consumption by taking into account: (i) the bandwidth required to communicate the safe region from the server to its corresponding mobile client, and (ii) the computation cost at a mobile client for monitoring its position with respect to the safe region.

In summary, safe region computation must satisfy the



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

Figure 1: Preliminaries for Safe Region Theorem

following constraints:

Lightweight Construction. Safe region computation should induce a low processing overhead at the server as this computation may need to be performed frequently for large number of mobile users.

Compact Representation. The safe region should have a compact representation as it needs to be communicated back to the user resulting in consumption of downstream bandwidth. A rectangular safe region is suitable in terms of meeting this requirement.

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

Device Heterogeneity. Mobile devices are known by their diversity in terms of resource capacity. Safe region computation techniques should be flexible and adaptive to computational heterogeneity of devices.

These requirements motivate us to design three alternative safe region techniques and study the impact of size and shape of safe region on server load, network bandwidth and client energy consumption.

2.2. Grid Overlay for Safe Region Computation

The goal of introducing safe region-based alarm evaluation is to enable the system to focus the processing on alarms that are in the vicinity of the mobile subscribers' current locations. In order to effectively support this objective, we use a grid structure overlaid on top of the Universe of Discourse considered by the system. Figure 3(a) shows the current grid cell of mobile user s. Though, using the current grid cell of a mobile user as its safe region is simple, when the number of intersecting spatial alarms is large, the use of this grid cell as safe region is too costly. Not only is the cost of communicating the large number of spatial alarms to the mobile user high, but the computational cost of the mobile client monitoring its position with respect to the safe region also increases dramatically. Therefore, we dedicate the next two sections to present algorithms that control the complexity of the size and shape of safe regions, making safe region-based processing truly attractive for scaling spatial alarm processing.

3. Maximum Weighted Perimeter Rectangular Safe Region

The safe region approach aims to reduce the number of alarm evaluations performed by the server. A rectangular shape has many properties required of the safe region as mentioned before. In this section, we discuss the *maximum weighted perimeter rectangular safe region* computation approach and present a safe region computation algorithm based on the concept of dynamic skylines [4].

Figure 1(a) displays a mobile user s at position $l_s(t)$; assume that $l_s(t')$ is the previously recorded position of the client. The probability density function (pdf) for the client motion inside the safe region, denoted by $p(\phi)$, is given by:

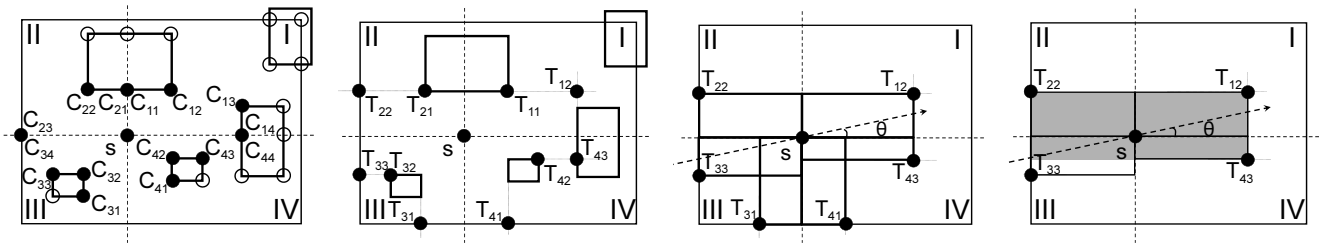
$$p(\phi) = \begin{cases} \dfrac{1 + \frac{y}{z}\left\lceil \frac{\pi/2 - |\phi|}{y/z \cdot \pi} \right\rceil}{2\pi} & \text{, if } -\pi/2 \leq \phi \leq \pi/2 \\ \dfrac{1 - \frac{y}{z}\left\lceil \frac{|\phi| - \pi/2}{y/z \cdot \pi} \right\rceil}{2\pi} & \text{, otherwise} \end{cases}$$

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

We now present an algorithm to compute the maximum weighted perimeter safe region for a user. Our algorithm applies the concept of *dominating point* and appropriate heuristics to find the *four skyline points* [5] which form the corner points of the rectangular safe region. The algorithm accepts the current position vector $\vec{l_s}$ for a user s and the current grid cell $\mathcal{G}(\vec{l_s})$ in which the user s resides as inputs. The set of relevant alarms intersecting the grid cell $\mathcal{G}(\vec{l_s})$ are considered for safe region computation. In case there are no relevant alarm regions intersecting the grid cell $\mathcal{G}(\vec{l_s})$, the entire cell is returned as the safe region. Otherwise, the algorithm proceeds in the following four steps outlined below. We refer readers to [6] for a complete description of the algorithm.

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

Concretely, the set of candidate points is determined as follows. First, the spatial region corner for each relevant alarm is selected as a candidate point in its appropriate



(a) Candidate Point Set (b) Tension Point Set (c) Component Rectangles (d) Final Safe Region

Figure 2: Maximum Weighted Perimeter Safe Region Computation

quadrant. The algorithm trims the set of candidate points in the next step. We remove points which *fully dominate*[1] any other point in $\mathcal{C}$. Finally, the points are sorted according to increasing distance of their x-coordinate from the origin.

Step 2: Determine Tension Point Set. Tension points are selected from the set of candidate points. In this step, we process the set of candidate points in the following manner to obtain the set of tension points. Each tension point T_{Qi}, where $Q \in \{1, 2, 3, 4\}$ represents the quadrant the point belongs to, is assigned the same x-coordinate as the corresponding candidate point C_{Qi}. T_{Qi} is assigned the same y-coordinate as that of C_{Qi-1}, or T_{Qi-1} if T_{Qi} and T_{Qi-1} have the same x-coordinate.

Step 3: Determine Component Rectangles. The set of tension points form the opposite corner (opposite to the origin) of the set of candidate *component rectangles* in each quadrant. The final safe region is composed of the intersection of the component rectangles from each quadrant.

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

Figure 2 shows an example of our safe region computation approach. The candidate point set (black dots) for the given scenario is as shown in Figure 2(a). Figure 2(b) displays the set of tension points obtained from the candidate point set as explained in Step 2. Figure 2(c) displays the component rectangles formed by selecting a few of the tension points. The shaded region, as shown in Figure 2(d), forms the final safe region.

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

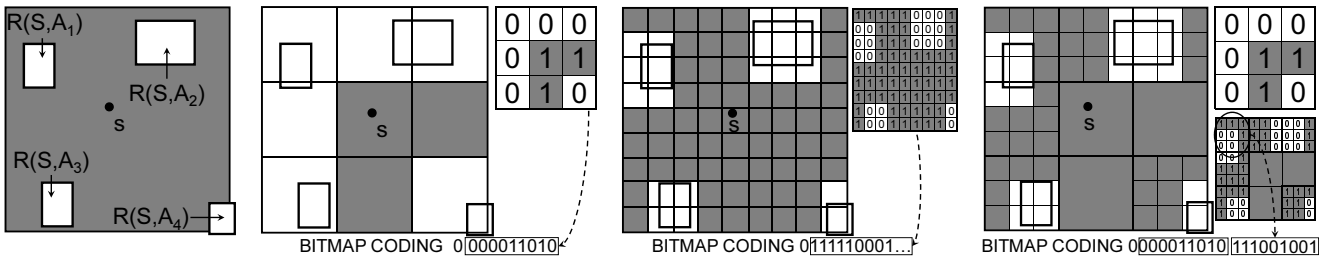
4. Bitmap Encoded Safe Region Computation

Compact representation of rectangular safe regions leads to low server-to-client communication cost and fast containment detection at the client end. However, the rectangular shape restriction is not optimal in terms of maximizing the safe region area. For mobile clients with high computational capacity, we can provide safe regions of finer granularity, more precisely, safe regions of larger size and more complex in shape.

In this section, we relax the rectangular shape restriction and consider rectilinear polygonal representations for safe region. We introduce *bitmap encoded safe region* (BSR) techniques for quickly and efficiently representing rectilinear polygons using bitmaps. This approach provides flexibility in safe region computation by providing larger, complex safe regions for powerful clients, thus personalizing the safe region for each client according to its computational capacity. Recall Figure 3(a), which displays the grid cell for subscriber s with four relevant intersecting alarm regions. One approach is to use the grid cell (minus spatial alarm regions) as the safe region for the client and communicate it to the client. This approach forces the server to communicate the current grid cell and all relevant alarms overlapping with the grid cell to the mobile client. Each alarm region may be represented by the bottom-left and top-right corner points. This approach can be considered *optimal* since it maximizes the size of the safe region by communicating to the client the complete knowledge of all alarms in its vicinity. However, this approach may be cost prohibitive in terms of server-to-client communication cost and computation cost incurred at the client in presence of large number of relevant alarms. For example, for areas with high alarm density the server may push a large number of alarms onto the client which would lead to heavy load for weak clients. To counter this problem, we develop the concept of *bitmap encoded safe regions*, which provides an estimation of the actual safe region using a bitmap, allowing for trade-off between the size of the bitmap and the accuracy of safe region representation.

Definition 1: A bitmap encoded safe region represents a safe region Ψ_s for subscriber s using a bitmap B. A bit value of 1 indicates that a predefined region (cell) belongs to the safe region; whereas a 0 bit indicates the negation.

We first describe a *Grid Bitmap Encoded Safe Region (GBSR)* computation technique and exhibit its inability to



(a) Grid Cell (b) Grid Bitmap Encoding (3×3) (c) Grid Bitmap Encoding (9×9) (d) Pyramid Bitmap Encoding

Figure 3: Bitmap Encoded Safe Region Computation

accurately and efficiently represent safe regions. An extension to this approach using the pyramid data structure [7], referred to as the *Pyramid Bitmap Encoded Safe Region (PBSR)* approach, allows us to represent safe regions accurately as well as efficiently. BSR techniques exhibit the following advantages: (i) for low alarm density regions, it allows for further reduction of alarm evaluations compared to the rectangular safe region approaches, (ii) it supports varying granularity of safe region computations thus supporting heterogeneity among client capabilities, and (iii) clients can determine their position with respect to the safe region using a predefined worst-case number of computations.

4.1. Grid Bitmap Encoded Safe Region

The safe region for a subscriber s can be represented by the set of grid cells as shown in Figure 3(b).

Proposition 2: We use a grid bitmap scheme to represent the safe region within the grid cell shown in Figure 3(a). The $\alpha \times \beta$ cell $C_{i,j}$ is represented by a single bit $B(C_{i,j})$. If $C_{i,j} \bigcap \Sigma_{k=1}^{m} R(s, A_k) = \emptyset$, we set $B(C_{i,j}) = 1$ denoting that the entire cell $C_{i,j}$ belongs to the safe region Ψ_s, else we set $B(C_{i,j}) = 0$ and split $C_{i,j}$ into $U \times V$ smaller equisized cells. The same encoding procedure is used for each smaller cell.

Figure 3(b) shows the safe region representation for the safe region of Figure 3(a) using a bitmap encoding scheme. No alarm regions intersect the three shaded cells which are represented by 1's; other cells intersecting with alarm regions are represented by 0's. The safe region is represented using a simple bitmap $B = 0000011010$ which represents the cell bit values in a raster scan fashion. The first zero bit corresponds to the entire cell, indicating that the cell does not belong to the safe region and has intersecting alarm regions. As visible from Figure 3(b), this bitmap encoding is able to represent only a small portion of the grid cell thus providing a poor estimate of the actual safe region. Figure 3(c) presents a 9×9 split of the cell at a finer resolution which allows for more accurate representation of the safe region. However, this approach is inefficient for the following two reasons: (i) it unnecessarily uses a much larger bitmap than required to represent the safe region, and (ii) different regions have different alarm densities thus making it difficult to select a uniform grid cell size. The PBSR approach overcomes these deficiencies by allowing for more accurate representations of the safe region while providing a smaller bitmap size.

4.2. Pyramid Bitmap Representation

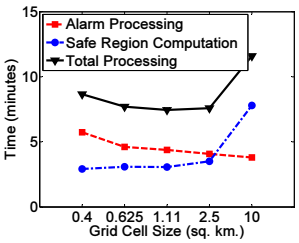
The pyramid representation splits cells in the *base* grid (level L=0) with $B(C_{i,j}^{L}) = 0$ *only* into $U \times V$ smaller cells, where $\alpha_U = \alpha/U$, $\beta_V = \beta/V$, where U, V are a system defined parameters. The process may be further repeated for several iterations to form smaller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

We omit the algorithm for safe region estimation using PBSR due to space constraints and refer interested readers to our technical report [6]. However, a brief outline of the procedure is given below. The pyramid representation of the base cells is constructed for height h by splitting cells iteratively into $U \times V$ cells. This step is performed offline by the server thus providing a precomputed pyramid representation for safe region computation. Next, starting from the base cells (level $L = 0$) we determine if each cell intersects any relevant alarms. Cells not intersecting with any relevant alarm regions are assigned a bit value $B(C_{i,j}^{L}) = 1$ indicating that they are a part of the safe region; else a cell is assigned bit value 0. For cells at each level $L-1$ ($L \leq h$) which have an assigned bit value 0, we consider the relevant $U \times V$ *children* cells at Level L and assign a bit value 0 or 1 considering intersection of the cell with relevant alarms at each level of the pyramid.

Proposition 3: The PBSR approach for safe region computation allows us to represent the safe region Ψ_s in terms of a bitmap of size $|B|$. The height of the pyramid h allows us

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

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



(b) Server Processing Time for Weighted Perimeter Approach (y=1, z=32)

Figure 4: Performance of Rectangular Approach

to control the accuracy of representation of the safe region at the cost of computing a larger bitmap for more accurate representations.

We define *Coverage* and *Bitmap Size* which allow us to control the quality of the safe region representation for our BSR computation techniques. The coverage of a safe region representation Ψ_s, denoted by $\eta(\Psi_s)$, is defined as the ratio of area of the safe region using the PBSR representation to the area of the grid cell. The bitmap size for safe region Ψ_s is defined as the number of bits in the PBSR representation of the safe region. In practice, we want to achieve high coverage with as small bitmap size as possible. Each client may specify the maximum height of the pyramid used by the PBSR approach for computing its safe region. In the worst case scenario, the client may need to determine its position relative to the safe region at each level of the pyramid data structure. We refer readers to our technical report [6] for a detailed description of safe region containment detection algorithm which performs pyramid bitmap decoding to obtain a geometrical shape of the safe region.

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

5. Experimental Evaluation

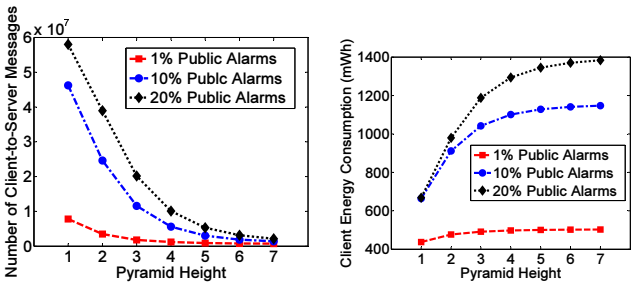
In this section, we evaluate the performance of our safe region computation techniques using three different sets of experiments. The first set of experiments performs an evaluation of the maximum weighted perimeter rectangular safe region approach. The second experiment evaluates the bitmap encoded safe region (BSR) approaches, namely grid bitmap encoded safe region (GBSR) and pyramid bitmap encoded safe region (PBSR). The final experiment provides an evaluation of the safe region techniques compared to periodic processing (PRD), safe period-based (SP) computation [3] and the optimal (OPT) approach as described in beginning of Section 4. The optimal approach does not consider any restrictions on resource availability and assumes all relevant alarms within the grid cell are pushed to the client, which implies the client is fully aware of all relevant alarms in its vicinity. We measure the performance of all approaches on different evaluation metrics like number of client-to-server messages, downstream server-client bandwidth consumption, client energy consumption and server processing time. The parameters adopted for each processing approach *ensure 100% of the alarms are triggered in all scenarios*. The sequence of alarms to be triggered is determined by a very high frequency trace of the motion pattern of the vehicles.

5.1. Experimental Setup

Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS [8]). The simulator simulates the motion of vehicles on roads with appropriate velocity information. We use a map of Atlanta and surrounding regions, which covers an area around 1000 km^2 in expanse, to generate the trace. Our experiments use traces generated by simulating vehicle movement for a period of one hour, results are averaged over a number of such traces. Default traffic volume values allow us to simulate the movement of a set of 10,000 vehicles on the above road network. Position parameters are evaluated against installed spatial alarms indexed in a R^*-tree [9]. The default spatial alarm information consists of a set of 10,000 spatial alarms installed on alarm targets distributed uniformly over the entire map region. Default setup assumes 10% of the alarms are public alarms; private and shared alarms are present in the system in the ratio 2:1.

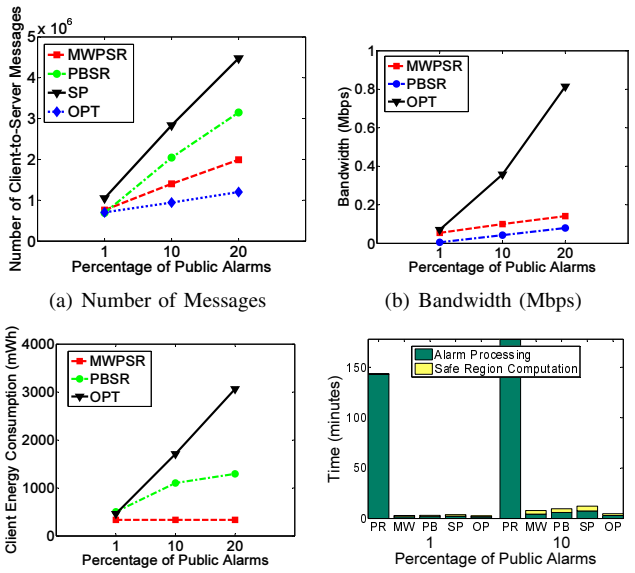
5.2. Experimental Results

Performance of Maximum Weighted Perimeter Rectangular Approach. This set of experiments is designed to study the performance of the maximum weighted perimeter rectangular safe region approach as we vary the parameters of steadiness y, z. The results for $y = 1$ and different values of z in comparison with a non-weighted approach (no steady motion assumption) are shown in Figure 4. The non-weighted perimeter approach improves upon the approach presented in [10] by allowing for overlapping alarm regions. The approach presented in [10] leads to alarm misses and erroneous safe regions in such scenarios. Figure 4(a) shows the number of client-to-server location messages exchanged with different grid cell sizes. The weighted perimeter approach consistently performs better than the non-weighted perimeter approach even though by a small margin. Considering the fact that more than 60 million location messages are produced for each trace, it



(a) Number of Messages (b) Client Energy Consumption

Figure 5: Performance of BSR Approach



(a) Number of Messages (b) Bandwidth (Mbps)

(c) Client Energy Consumption (d) Server Processing Time

Figure 6: Safe Region vs. Other Approaches

can be observed that less than 3% of messages need to be communicated to the server using any of the rectangular safe region approaches. The other observation from this figure is that with increasing grid cell size the number of client-to-server messages reduces. This is as expected because with larger grid cell sizes larger safe regions are computed and the client stays within the safe region for a longer duration. Figure 4(b) shows the server processing time as we vary the size of the grid cell. As grid cell size is increased, alarm processing costs decrease due to the smaller number of location messages being processed against the alarm index. The safe region computation costs increase with increasing grid cell size due to larger number of intersecting alarms being considered for safe region computation. The total server processing time is minimum with a grid cell size of 2.5 km^2.

Performance of BSR Approach. This set of experiments is designed to evaluate the performance of the BSR approach. We vary the height of the pyramid from $h = 1$ (for GBSR) to $h = 7$ and observe the performance as shown in Figure 5. Figure 5(a) displays the number of client-to-server messages communicated as we increase the pyramid height from $h = 1$ to $h = 7$. It can be observed that the GBSR approach is highly inefficient as it limits safe region computation to a very high granularity. The safe region computed using this approach provides a very coarse representation of the actual safe region forcing the clients to frequently send location messages as a result of which GBSR approach incurs high communication costs. As we increase the pyramid height, more accurate safe region representations can be computed and consequently number of messages transmitted experiences a sharp drop. Another observation is that BSR approaches display high sensitivity to alarm density levels; the performance deteriorates sharply for higher percentage of installed public alarms. Figure 5(b) displays the client energy consumption (in milliwatt-hours) used to determine client position within the safe region. We omit details related to energy consumption calculations due to space constraints. For the GBSR approach the clients need to perform an average of 2-3 safe region containment detections per second resulting in low energy consumption. This cost does not experience a significant increase with pyramid height for low percentage of public alarms. For higher public alarm percentages, 6-7 safe region containment detections per second are required for a pyramid of height $h = 7$ resulting in higher energy consumption.

Performance Comparison of Safe Region with Other Approaches. Now we compare the performance of the maximum weighted perimeter rectangular safe region approach (MWPSR) and the pyramid bitmap safe region (PBSR) approach ($h = 5$) with periodic evaluation (PRD), safe period-based processing (SP) and the optimal approach (OPT) as described at the beginning of Section 4. As can be seen from Figure 6(a), the safe region approaches transmit few client-to-server messages. Periodic processing requires clients to transmit each location update to the server amounting to 60 million messages and is not shown in the figure. The safe period approach experiences significantly higher communication costs, approximately 2-3 times the cost incurred by the safe region approaches. This is due to the pessimistic assumptions required to ensure that the safe period approach triggers all alarms with a 100% success rate. The optimal approach would require clients to transmit updates only when the spatial constraints for one or more relevant alarms are met and transmits fewest number of messages. Figure 6(b) displays the downstream bandwidth consumed by the system to broadcast safe regions to the clients. Safe period approach would also require that a computed safe period be broadcast to each client; however, we exclude the bandwidth incurred for this approach from these results. As expected the safe region approaches incur much lower bandwidth expense when compared to an optimal solution. PBSR ($h = 5$) performs the best for different percentage of public alarms in the system. Not surprisingly, client energy consumption for the optimal approach is significantly higher than the safe region approaches (Figure 6(c)) as the optimal solution is based on the assumption that clients have very high capacity.

PBSR and MWPSR approaches lead to lower client energy consumption especially at higher alarm density levels. The processing load experienced by each approach is as shown in Figure 6(d). Periodic approach (PR) has much higher alarm processing costs as each update needs to be processed by the client and the server load does not scale. The processing load does not rise much at higher alarm densities as each update is processed by this approach for all percentage of public alarms. The MWPSR and PBSR approaches (MW and PB in the figure) experience lower server load due to much lower alarm processing load. With increasing percentage of public alarms, the safe region computation as well as the alarm processing load rises; however, the total load incurred by the system is much lower than the periodic approach for all configurations. The PBSR approach again shows similar trends as the MWPSR approach; however, the CPU load incurred by this approach at higher alarm density levels is higher than MWPSR approach. The safe period (SP) approach experiences higher CPU load compared to the safe region approaches. This is a direct result of the larger number of updates that need to be processed by the safe period approach. Results for the optimal approach (OP) are plotted to show that the safe region approaches do not incur much higher CPU load except for the highest percentage of public alarms.

6. Related Work

In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand [11], [12]. In addition to monitoring continuously changing user information needs, spatial alarm processing systems also need to deal with the complexity of monitoring user location data in order to trigger relevant alerts in a non-intrusive manner.

Periodic reevaluation is commonly used for continuous monitoring of moving objects [13], [14], [15], [16]. Some work exists on monitoring continuous queries which applies the concept of safe region directly [17], [10], [15] or indirectly [18], [19]. Spatial alarms differ from this work as they do not demand periodic evaluation like continuous queries; instead they require one shot evaluation which should result in a trigger when the alarm conditions are satisfied. Our work is focused on determining the opportune moment for evaluating spatial alarms relevant to a client by seeking cooperation at the client end.

None of the previous work, except [10], presents clear algorithms for safe region computation. The maximum weighted perimeter rectangular safe region approach outperforms the approach presented in [10]. Further, unlike our approach, the algorithm presented in [10] cannot handle overlapping alarm regions or alarm regions intersecting the axes of the coordinate system. Most importantly, previous work fails to consider an environment supporting heterogeneous clients. Our PBSR technique provides an elegant solution for exploiting client heterogeneity further easing the computational load on the server.

7. Conclusion

In this paper, we have presented a distributed safe region-based architecture for scaling spatial alarms-enabled services. The paper makes three important contributions. First, we introduce the concept of safe region-based alarm processing to enhance system scalability. Second, we develop alternative techniques for safe region computation: maximum weighted perimeter rectangular safe region and the grid and pyramid-based bitmap encoded safe region approaches. Third, we consider trade-offs between different safe region computation techniques and explore their impact on the client-to-server communication cost, server load, downstream bandwidth consumption and client energy consumption. Our experimental evaluation demonstrated the advantages of our distributed safe region-based approach in comparison with server-centric alarm processing techniques such as periodic evaluation and safe period-based evaluation.

Acknowledgment

This work is partially supported by grants from NSF CISE CyberTrust, SGER, IBM faculty award, IBM SUR program, and an Intel University Research Grant.

References

[1] "GPS-Enabled Mobile Devices Market Research Report," http://www.abiresearch.com/products/market_research/GPS_Enabled_Mobile_%Devices.
[2] A. Murugappan and L. Liu, "Energy-Efficient Processing of Spatial Alarms on Mobile Clients," in *SEDE*, 2008.
[3] B. Bamba, L. Liu, P. S. Yu, G. Zhang, and M. Doo, "Scalable Processing of Spatial Alarms," in *IEEE HiPC*, 2008.
[4] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," in *SIGMOD*, 2003.
[5] K. Tan, P. Eng, and B. Ooi, "Efficient Progressive Skyline Computation," in *VLDB*, 2001.
[6] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu, "Safe Region Techniques for Fast Spatial Alarm Evaluation," Georgia Institute of Technology, Tech. Rep., 2008.
[7] H. Samet, *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Reading, Mass, 1990.
[8] "U.S. Geological Survey," http://www.usgs.gov.
[9] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," in *SIGMOD*, 1990.
[10] H. Hu, J. Xu, and D. Lee, "A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects," in *ACM SIGMOD*, 2005.
[11] 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.
[12] L. Liu, C. Pu, and W. Tang, "WebCQ - Detecting and Delivering Information Changes on the Web," in *CIKM*, 2000.
[13] C. Jensen, D. Lin, and B. Ooi, "Query and Update Efficient B+-Tree based Indexing of Moving Objects," in *VLDB*, 2004.
[14] M. Mokbel, X. Xiong, and W. Aref, "SINA: Scalable Incremental Processing of Continuous Queries in Spatio-Temporal Databases," in *ACM SIGMOD*, 2004.
[15] 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*.
[16] X. Yu, K. Pu, and N. Koudas, "Monitoring k-Nearest Neighbor Queries over Moving Objects," in *ICDE*, 2005.
[17] Y. Cai and K. Hua, "An Adaptive Query Management Technique for Efficient Real-Time Monitoring of Spatial Regions in Mobile Database Systems," in *IEEE IPCCC*, 2002.
[18] J. Xu, X. Tang, and D. Lee, "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," *IEEE TKDE*, 2003.
[19] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. Lee, "Location-based Spatial Queries," in *ACM SIGMOD*, 2003.