Scaling Spatial Alarm Services on Road Networks

Kisung Lee, Ling Liu, Shicong Meng, Balaji Palanisamy
DiSL, College of Computing, Georgia Institute of Technology, Atlanta, USA
{kisung.lee, lingliu, smeng, balaji}@cc.gatech.edu

***Abstract*—Spatial alarm services are essential components of many location-based applications. One of the key technical challenges for supporting spatial alarms as a service is performance and scalability. This paper shows that the Euclidean distance-based spatial alarm processing techniques are inadequate for mobile users traveling on road networks due to the high overhead in terms of server load for alarm checks and the high energy consumption in terms of client wakeups. We design and develop ROADALARM, a road network aware spatial alarm processing service, with three unique features. First, we introduce the concept of road network-based spatial alarms using road network distance measures and a set of metrics specialized for spatial alarm processing. Second, we develop the basic model for spatial alarm processing by exploiting two types of filters: subscription filter and Euclidean lower bound filter. Third and but not the least, we develop a suite of optimization techniques to further reduce the frequency of wakeups at mobile clients and the number of alarm checks at the alarm processing server, while ensuring high accuracy of spatial alarm processing. Our experimental results show that ROADALARM outperforms existing Euclidean space-based approaches with high success rate (accuracy) and significantly increased hibernation time.**

I. INTRODUCTION

Ubiquitous connectivity and cloud computing are enabling technologies that hold the promise of offering many new Web services and content-rich applications. Spatial alarm services represent a new class of emerging location-aware, just-in-time Web services, which can disseminate information of interest to the right users at the right time and the right place.

Spatial alarms extend the concept of time-based alarms to spatial dimension and remind us when we enter some predefined location of interest in the future. For example "alert me when I am within 2 miles of the dry clean store at the junction of Druid Hill and Baircliff" is a personalized spatial alarm. Spatial alarms are basic building blocks for many location-based services, such as location-based advertisements, location-based social networks, location-based tourism and entertainment, or highway traffic alert systems. For example, when installing applications of popular restaurants on smart-phones, many mobile users are interested in those applications that provide discount coupons of nearby restaurants based on their current locations.

A spatial alarm is defined by four components: a focal point representing the alarm target, a spatial distance representing the alarm region, an alarm publisher and a set of alarm subscribers. Spatial alarms are categorized into three groups: *private*, *shared* and *public*. A *private* alarm has only one subscriber who is also its publisher. A *shared* alarm has a publisher and several subscribers approved by the publisher. In terms of a *public* alarm, its publisher does not set any restriction on subscribers and thus anyone can be a subscriber of the alarm. Public alarms are typically classified by alarm interests, such as traffic alerts, coupons from grocery stores.

With a rapidly increasing number of smart-phones, one of the most important technical challenges for delivering spatial alarms as a service is scalability and performance in terms of both server computation time and client energy consumption. First, we argue that negligent management of spatial alarms can lead to unnecessary consumption of energy at mobile devices, especially those with limited battery power because continuous tracking and notification is known to be costly for battery life at mobile devices. According to [1], minimizing use of location services is listed as the first tip to extend smart-phones' battery life. Furthermore, the performance of spatial alarm processing can be affected by a number of factors, such as *frequency of wakeups* − how often mobile devices should wake up because of possible alarm hits and *frequency of alarm checks* − how many spatial alarms should be evaluated at each wakeup. Since frequent and often unnecessary wakeups and alarm checks at each wakeup not only reduce battery life of mobile devices considerably but also increase the loads of a spatial alarm processing server, we need techniques for efficient processing of spatial alarms, which can reduce both the number of unnecessary wakeups and the number of excessive alarm checks at each wakeup. At the same time, we need to guarantee that the alarm processing service can scale to a large number of spatial alarms and a growing number of mobile users, while meeting the *high accuracy* requirement by minimizing the alarm miss rate.

Existing approaches on spatial alarm processing are categorized into two groups by their methods of controlling the frequency of wakeups. The first category is called *time-based* approaches, which periodically perform alarm checks at the server and inform mobile subscribers about the alarm check results. The second category is referred to as *distance-based* approaches, which compute the distance from a mobile subscriber to each of its subscribed alarms based on regular or irregular time interval and inform the mobile subscribers when they approach an alarm target of their subscribed spatial alarms. Too frequent alarm checks can cause excessive loads on alarm servers and unnecessary wakeups on mobile clients. However, less frequent alarm checks may cause high alarm miss rate. The state of art techniques to spatial alarm processing is safe period [3] and safe region [2], [5] techniques, which use Euclidean distance between a mobile subscriber and its closest alarm to determine the safe region or safe

period to move without checking alarms. Surprisingly, no existing research has tried to capitalize on spatial and mobility constraints of mobile devices traveling on road networks for optimizing and scaling spatial alarm services. We argue that by developing road network aware spatial alarm processing techniques, one can offer spatial alarm services with both high performance and high accuracy.

In this paper, we present ROADALARM – a road network aware spatial alarm processing service architecture and a suite of road network aware techniques for scaling spatial alarm processing in terms of a growing number of mobile users and alarm subscriptions. By taking into account spatial constraints on road networks and mobility patterns of mobile users, the ROADALARM approach can filter out those spatial alarms that are irrelevant or far away from the current location of their mobile subscribers. We have developed three types of alarm filters with two objectives: (i) to reduce the frequency of wakeups and increase the hibernation time of mobile clients (saving battery and enhancing service usability), and at the same time, (ii) to minimize the computation cost of calculating the hibernation time for a mobile client at each wakeup and evaluating all of its subscribed alarms. Concretely, instead of a rectangle region, we define a road network-based spatial alarm as a star-like subgraph with an alarm target as the center of the star and a road network distance as a safe hibernation measure. We define the scope of an alarm region by the set of border points of the star. By exploiting subscription filtering and Euclidean lower bound filtering, we formulate the basic model of ROADALARM for processing road network aware spatial alarms. Furthermore, we develop a suite of motion-aware filters, which can utilize the mobility information to further reduce the frequency of wakeups at mobile clients and alarm checks at the server, while ensuring high accuracy of alarm evaluation. To the best of our knowledge, ROADALARM is the first systematic approach to exploring road network aware and motion aware filters to reduce the search space and computation cost of spatial alarm processing. We conduct extensive experimental evaluation and our results show that the ROADALARM approach significantly outperforms existing Euclidean distance-based techniques and can scale to a growing number of spatial alarms as well as mobile subscribers.

II. ROADALARM SERVICE: AN OVERVIEW

A spatial alarm service typically consists of a spatial alarm processing engine and a location server where the locations of mobile users and static objects (such as gas stations, restaurants, and so on) are managed. The spatial alarm processing engine communicates with the location server to obtain the current road network locations of mobile subscribers as well as alarm targets for all alarms maintained in its database. The location server uses localization techniques (such as GPS, WiFi or any hybrid localization technology) to keep track of the current positions of mobile users. The spatial alarm processing engines can be geographically distributed and register spatial alarms of interest, and synchronize with the location server to continuously monitor the spatial alarms of mobile subscribers as they move on the road. Fig. 1(a) presents a sketch of the ROADALARM service architecture.

We assume that mobile clients can be any devices (e.g. smart-phones, navigation systems) with any localization technology such as GPS and WiFi localization. ROADALARM adopts the client-server architecture for spatial alarm processing. Concretely, mobile users may install (publish) their spatial alarms at the location server as private, shared or pubic alarms. In addition to their own private alarms, mobile users can subscribe any public alarms of their interest and a subset of shared alarms authorized by the respective alarm publishers. Mobile users need to install the thin client of ROADALARM as a mobile application on their devices. Each mobile subscriber will obtain an initial hibernation time at the commit of her alarm subscription. Upon the expiration of its old hibernation time, the mobile client will automatically contact the alarm server to obtain its new hibernation time. During the hibernation time, the ROADALARM application is hibernated at the mobile client, and the alarm server consumers zero alarm processing cost for this mobile client.

A. Road network reference model

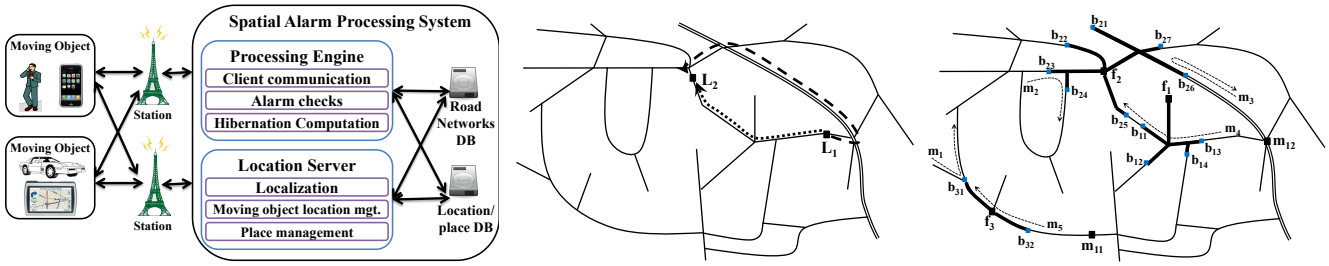
The road network is represented by a directed graph $G = (\mathcal{V}, \mathcal{E})$, composed of the junction nodes $\mathcal{V} = \{n_0, n_1, \ldots, n_N\}$ and directed edges $\mathcal{E} = \{n_i n_j | n_i, n_j \in \mathcal{V}\}$. We refer to an edge $n_i n_j$ as a road segment connecting the two end nodes n_i and n_j with direction from n_i to n_j. When a road segment is bidirectional, we use edge $n_i n_j$ and edge $n_j n_i$ to denote the two directions of the same road segment with n_i and n_j as the starting nodes respectively. For each road segment, road-related information can be maintained, such as segment length (e.g. 1.2 miles), speed limit (e.g. 55 mph), current traffic data (e.g. average speed is 35 mph), direction (e.g. one-way road), etc. The length and speed limit of a road segment $n_i n_j$ are denoted by $seglength(n_i n_j)$ in miles and $speedlimit(n_i n_j)$ in miles per hour respectively. Other road-related information such as direction and current traffic data, if available, can be easily incorporated to provide more accurate travel time.

Let n_1 and n_2 denote two road junction nodes and $n_1 n_2 \notin \mathcal{E}$. We define a path from a node junction n_1 to a node junction n_2 as a sequence of road segment edges, one connected to another, denoted as $n_1 n_{i_1}$, $n_{i_1} n_{i_2}, \ldots, n_{i_{k-1}} n_{i_k}$, $n_{i_k} n_2$ ($k > 0$). The length of a path h between n_1 and n_2 is computed as follows:

$$pathlength(h) = seglength(n_1 n_{i_1}) + seglength(n_{i_k} n_2) + \sum_{\alpha=1}^{k-1} seglength(n_{i_\alpha} n_{i_{\alpha+1}})$$

Given two road junctions n_1 and n_2, since there are more than one way from n_1 to n_2, we use $PathSet(n_1, n_2)$ to denote the set of all paths between n_1 and n_2. We define a segment length-based shortest path between n_1 and n_2, denoted by $sl\_shortestpath(n_1, n_2)$, as follows:

$$\{h_{sl} | pathlength(h_{sl}) = \min_{h \in PathSet(n_1, n_2)} pathlength(h)\}$$



(a) Service Architecture (b) Segment length-based vs Travel time-based (c) Road Network-based Spatial Alarms

Fig. 1. Spatial Alarm Services on Road Networks

The travel time of a road segment $n_i n_j$ is $\frac{seglength(n_i n_j)}{speedlimit(n_i n_j)}$ and thus the travel time of a path h is calculated as follows:

$$pathtime(h) = \frac{seglength(n_1 n_{i_1})}{speedlimit(n_1 n_{i_1})} + \frac{seglength(n_{i_k} n_2)}{speedlimit(n_{i_k} n_2)} + \sum_{\alpha=1}^{k-1} \frac{seglength(n_{i_\alpha} n_{i_{\alpha+1}})}{speedlimit(n_{i_\alpha} n_{i_{\alpha+1}})}$$

The travel time-based shortest path between n_1 and n_2, denoted by $tt\_shortestpath(n_1, n_2)$, is defined as follows:

$$\{h_{tt} | pathtime(h_{tt}) = \min_{h \in PathSet(n_1, n_2)} pathtime(h)\}$$

A *road network location*, denoted by $L = (n_i n_j, p)$, is a tuple of two elements: a road segment $n_i n_j$ and the *progress* p along the segment from n_i to n_j. The road network distance between two road network locations $L_1 = (n_{i_1} n_{i_2}, p_1)$ and $L_2 = (n_{j_1} n_{j_2}, p_2)$ is the length of the shortest path between L_1 and L_2 in terms of either segment length or travel time. The **segment length-based road network distance** and **travel time-based road network distance** are formally defined respectively as follows:

$$sldistance(L_1, L_2) = seglength(n_{i_1} n_{i_2}) - p_1 + p_2 + pathlength(sl\_shortestpath(n_{i_2}, n_{j_1}))$$

$$ttdistance(L_1, L_2) = \frac{seglength(n_{i_1} n_{i_2}) - p_1}{speedlimit(n_{i_1} n_{i_2})} + \frac{p_2}{speedlimit(n_{j_1} n_{j_2})} + pathtime(tt\_shortestpath(n_{i_2}, n_{j_1}))$$

Even though the segment length-based distance is the most commonly used distance measure on road networks, it may not provide sufficient and accurate distance information in terms of actual travel time from the current location (L_1) to the destination (L_2). For instance, highway road segments are much longer but also with much higher speed limits and thus may have relatively lower travel time compared to some local road segments, as shown in Fig. 1(b). To ensure high accuracy and high performance of spatial alarm processing, in RoadAlarm we use the travel time-based distance as default road network distance measure.

B. Road Network-based Spatial Alarms

In the RoadAlarm service, we define a spatial alarm as a star-shaped subgraph centered at the focal point of the alarm target, denoted as $SA(f, r, S)$ where f is the focal point of alarm target (a road network location), r is the alarm monitoring region, represented by a spatial range (segment length or travel time) from f, and S is a set of subscribers. For each road network spatial alarm SA, we compute the set of *border points* which bound the star-shaped spatial alarm SA, denoted by $BorderPoints(SA) = \{b_1, b_2, \ldots, b_m\}$, such that b_i $(1 \leq i \leq m)$ is a network location and $sldistance(f, b_i) = r$. Consider Fig. 1(c) that shows three star-shaped alarms with focal points f_1, f_2 and f_3. b_{12} is one of the four border points of the alarm with focal point f_1.

We define a *hibernation time* for each mobile user, which is a time interval during which the mobile client does not need to wake up and the alarm server does not need to perform alarm checks for this mobile subscriber. A hibernation time is specified by a time interval consisting of its start time and end time. If the current time is between them, the mobile client's current status is *hibernation*; otherwise, it is *alive*. The timing of alarm triggering is very important for alarm evaluation accuracy. For example, based on the current hibernation time for the mobile user m_5 in Fig. 1(c), m_5 may receive the spatial alarm alert when it approaches the focal point f_3 or just before leaving the spatial alarm target region through the border point b_{31}. Thus, in RoadAlarm we use a stronger definition of alarm miss: if a mobile user's current status is *hibernation* when it enters alarm region of a spatial alarm by crossing a border point of the alarm, we treat it as an alarm miss. The *alarm success rate* is the percentage of spatial alarm alerts which are not missed. For example, if there are 9 alarm hits and 1 alarm miss (actual hit but not triggered), the success rate is 90%.

III. RoadAlarm Basic Algorithm

The most intuitive approach to implementing spatial alarms is the **Euclidean distance-based** approach. Concretely, for every mobile user m, upon the expiration of its hibernation time, m wakes up and contacts the spatial alarm server to obtain its new hibernation time. At the server, we first retrieve the index of all spatial alarms and obtain the set of border points for each active alarm. Then we compute the Euclidean distance between the current location of m (L_m) and each of the border points for all alarms and select the border point that is nearest to L_m to compute the shortest distance from m to the alarm target region. According to this shortest distance and a velocity metric of m, such as the global maximum speed or the

expected speed of m [3], we calculate the new hibernation time for m. Though this approach is easy to implement, its main weakness is the unnecessarily short hibernation time due to the use of Euclidean distance rather than road network distance. Consequently, mobile clients need to wake up frequently, consuming higher battery energy than necessary.

Another intuitive approach to evaluating road network-based spatial alarms is to use Dijkstra's **network expansion** algorithm [4]. When a mobile client m wakes up, the network expansion-based approach first retrieves a set of spatial alarms (A_m) subscribed by m. For each spatial alarm $a_i \in A_m$, we first obtain the set of border points of a_i. For each border point of a_i, we calculate the shortest path using Dijkstra's network expansion algorithm. By examining the border points of all alarms in A_m, we select the shortest path with the minimum travel time to compute the new hibernation time for m. The drawback of the network expansion approach is the high cost for the large number of shortest path computations since it needs to compute the shortest path from the current location of m to each of the border points of every spatial alarm subscribed by a mobile user m at each wakeup.

In this paper we first present the ROADALARM basic algorithm based on two alarm filters. We use the subscription filter to scope the computation of the hibernation time for each mobile user m to only those alarms that are subscribed by m. In addition, we use *Euclidean lower bound (ELB)* as the second type of filter, which further reduces the excessive shortest path computations by filtering out some irrelevant border points for each alarm selected by the subscription filter. The concept of *Euclidean lower bound* refers to the fact that the segment length-based shortest path distance between two network locations L_1 and L_2 is at least equal to or longer than the Euclidean distance between L_1 and L_2. By combining subscription filter and ELB filter, the ROADALARM basic approach outperforms both the Euclidean distance-based approach and network expansion-based approach by minimizing the shortest path computations required for computing hibernation time for each mobile subscriber while maintaining the accuracy of alarm evaluation.

Concretely, instead of computing shortest paths from the current location L_m of the mobile user m to every border point of all alarms subscribed by m, the ROADALARM basic approach computes the new hibernation time of m in five steps: (1)For every alarm subscribed by m, denoted by $a_i \in A_m$, we find the border point that has the shortest distance from L_m. Instead of computing shortest paths from L_m to every border point of alarm a_i, we compute the Euclidean distance between L_m and every border point of a_i and sort the set of border points based on their Euclidean distances to L_m in an ascending order using the Incremental Euclidean Restriction (IER) algorithm [6], [11]. (2) Let b_{nn} denote the border point that has the smallest Euclidean distance to L_m. We compute the segment length-based shortest path between L_m and b_{nn}. (3) We use a binary search algorithm to examine the sorted list of border points and remove those border points whose Euclidean distance to L_m is bigger than *sldistance*(b_{nn},L_m). (4) For each remaining border point b_j, we compute the shortest path between b_j and L_m. If *sldistance*(b_j,L_m) < *sldistance*(b_{nn},L_m) holds, we assign b_j to be b_{nn}. Thus, for a given mobile user and an alarm $a_i \in A_m$, the nearest border point b_{nn} of a_i will be used as the reference border point of a_i to compute the hibernation time for m. (5) Finally, the ROADALARM basic approach examines every alarm $a_i \in A_m$ and its nearest border point b_{nn} and chooses the border point whose segment length-based distance from L_m is the smallest. Let b_{min} denote this nearest border point and p_{min} denote the shortest path from L_m to b_{min}. Thus we compute the new hibernation time for m using the travel time of p_{min}.

Now we illustrate the working of the ROADALARM basic approach using the example in Fig. 1(c). We have three spatial alarms a_1, a_2, a_3 with focal points f_1, f_2 and f_3 respectively and two mobile users m_{11} and m_{12}. m_{11} subscribes to a_1 and a_3 and m_{12} subscribes to a_1 and a_2. Let $L_{m_{11}}$ and $L_{m_{12}}$ denote the current location of m_{11} and m_{12} respectively. When m_{11} and m_{12} wake up upon the expiration of their hibernation time, without subscription filter and ELB filter, we will need to compute the shortest path from $L_{m_{11}}$ and $L_{m_{12}}$ to all 13 border points and then choose the nearest border point to m_{11} and m_{12} respectively. With the subscription filter, we can filter out alarm a_2 for m_{11} and alarm a_3 for m_{12} when computing the new hibernation time. By ELB filter, to find the new hibernation time for m_{12}, we only need to perform one shortest path computation between $L_{m_{12}}$ and b_{13}. This is because by Euclidean distance, b_{13} is the nearest border point of a_1 to $L_{m_{12}}$ and b_{26} is the nearest border point of a_2 to $L_{m_{12}}$. Given that *eucdistance*($L_{m_{12}}$,b_{13}) < *eucdistance*($L_{m_{12}}$,b_{26}), b_{13} is the nearest border point for m_{12}. Now we compute the segment length-based network distance between $L_{m_{12}}$ and b_{13}, denoted by *sldistance*($L_{m_{12}}$,b_{13}). By comparing it with the Euclidean distance from $L_{m_{12}}$ to all other border points of a_1 and a_2, we find that the following condition *eucdistance*($L_{m_{12}}$,b_k) > *sldistance*($L_{m_{12}}$,b_{13}) holds ($k = 11, 12, 14, 21, 22, 23, 24, 25, 26, 27$). Thus we effectively removed 10 unnecessary shortest path computations.

IV. MOTION-AWARE OPTIMIZATIONS

Comparing to the network expansion-based approaches, the ROADALARM basic approach improves the efficiency of spatial alarm processing by using the ELB filter. This leads to removing a fair number of shortest path computations that are unnecessary for computing the new hibernation time for each mobile user. However, the ELB filter is not always effective. In some cases, the number of border points after applying ELB filter remains high. Recall the case of m_{11} and alarm a_1 in Fig. 1(c), the Euclidean distance from m_{11} to b_{12} is the shortest, thus the road network-based distance from m_{11} to b_{12} is first calculated. Because this distance is longer than the Euclidean distances from m_{11} to all other border points (b_{11}, b_{13}, b_{14}), the ELB filter filters out none of the border points for alarm a_1.

In this section we introduce motion-aware filters to further reduce the search space and the computation time of

ROADALARM basic approach, especially for mobile users subscribed many alarms and their alarms are scattered in a large geographical area. The main idea of the motion-aware filters comes from the observation that mobile users traveling on road networks typically exhibit some degree of steady motion. First, a mobile user traveling on a road network can move only by following the predefined road segments connected to the current road segment it resides. For example, if a mobile user is marching on a road segment, its current moving direction cannot be changed until it reaches a road junction. Furthermore, even if it reaches a road junction, it has high probability to follow the major road segment in the same or similar direction at the junction node. We refer to such motion behavior as steady motion. In the subsequent sections, we present three types of steady motion-based filters and discuss how each may reduce the search space and the computation time of the ROADALARM basic approach.

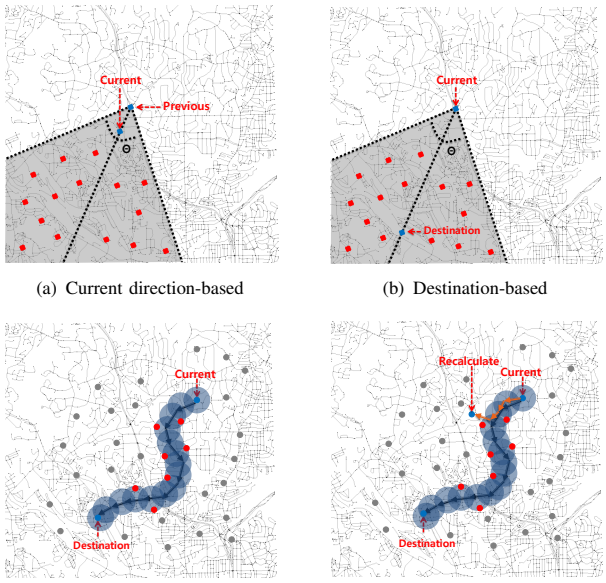
A. Steady Motion-based Filtering

The steady motion assumption in mobile computing systems refers to the fact that mobile users on the road will move along its current direction for a certain period of time. Formally, we can model the steady motion as follows: Let $p(\phi)$ denote the probability density function (PDF) for a degree ϕ based on the current direction of a mobile user m in the Euclidean space. $p(\phi)$ is defined based on the setting of ϕ and how steady the mobile user moves along the current direction on the road. The more steady the mobile user moves along the current direction and the smaller the ϕ value is, the higher probability $p(\phi)$ will be and the more effective the steady motion-based optimization will be in terms of filtering out irrelevant border points and alarms while maintaining high accuracy of alarm evaluation. We define $p(\phi)$ as follows:

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

where s/t represents the steadiness such that $s/t < 1$. For example, if s/t is equal to $1/2$, $P\left[-\pi/2 \le \phi \le \pi/2\right] = 0.75$ which means that the probability of moving within 180 degrees based on the current direction is 75%. On the other hand, if s/t is equal to zero, it means there is no steadiness and thus the probability is $1/2\pi$ for all values of ϕ.

Our optimizations use steady motion degree Θ to capture the constrained motion characteristics of mobile users traveling on a road network. For each mobile user, its steady motion degree Θ is utilized to limit the search space where it has high probability to be visited by the mobile user, based on the above PDF. If a sharp turn occurs at a junction node, a new Θ value will be computed for the mobile user based on the characteristics of underlying road networks and past movement history of the mobile user. We can also view this



(a) Current direction-based (b) Destination-based

(c) Shortest path-based (d) Shortest path recalculation

Fig. 2. Motion-aware Optimizations

Θ as a confidence indicator. When a mobile user moves on the road network by following its current direction, a large Θ value indicates possible sharp turns and sudden travel direction changes whereas a small Θ value indicates high probability of steady motion along the current direction. When a mobile user is traveling on the road network with a clear destination in mind, this Θ angle can be determined based on the current location of the mobile user and the destination location.

B. Direction-based Motion-aware Filter

The first motion-aware filter is based on the current direction of mobile users and their steady motion degree Θ. The Θ region is determined based on the current travel direction of the mobile user. Let (p_1, p_2) and (c_1, c_2) denote the previous location and the current location of a mobile user m respectively. The current travel direction vector of m is defined as $v = <c_1 - p_1, c_2 - p_2>$. Based on this vector, when a mobile user m wakes up, this filter limits the search space using the steady motion degree Θ and selects only those border points that reside in the Θ region anchored at the previous location of m as shown in Fig. 2(a). Concretely, let (xb_1, xb_2) denote a border point of some alarms subscribed by m. To check if the border point is within the Θ reduced search space, this filter first defines another vector $w = <xb_1 - p_1, xb_2 - p_2>$ and then calculates the degree of the border point from the current direction vector v using the following equation:

$$\textit{sm\_degree}(v, w) = \arccos\left(\frac{v \cdot w}{|v||w|}\right)$$

If $\textit{sm\_degree}(v, w) > \Theta$, this border point is removed since it is outside the constrained search space. For the selected border points, we calculate the new hibernation time by executing the ROADALARM basic algorithm, which uses the border point that is closest to the current location of mobile user m.

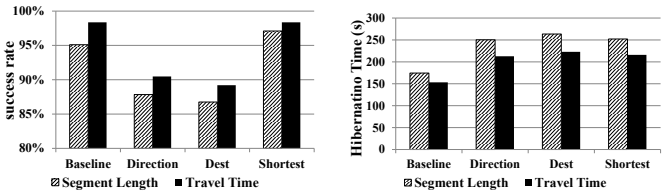
C. Destination-based Motion-aware Filter

The destination-based motion-aware filter utilizes both the current location and the destination information of mobile users. Destination information can be directly given by the mobile clients, such as those using car navigational systems or can be extracted from mobile clients' calendar applications. The destination-based filter chooses only border points which reside in the Θ region defined based on the current location of mobile users and their destination. We define a *destination vector* to represent the direction toward the destination, in which the current location and the destination location are used as the initial and terminal point of the vector respectively. When m wakes up, the destination-based filter restricts the search space using the destination vector and Θ. Only the border points of m that are located within this Θ restricted search space are selected as shown in Fig. 2(b). For the selected border points, we calculate the new hibernation time for m by executing the ROADALARM basic algorithm.

D. Shortest Path-based Motion-aware Filter

Even though both the current direction-based filter and the destination-based filter can reduce the computation cost of finding the nearest border point for each mobile user upon its wakeup by reducing the number of candidate border points and thus the search space, it still needs to examine too many border points in order to find the nearest one, especially when Θ is large and many alarms are subscribed by mobile users. Consider Fig. 2(b): the border points on the bottom far left or far right corner are unlikely to be hit by the mobile user since it is far away from the user's destination. Motivated by this observation, we propose the shortest path-based motion-aware filter based on a natural assumption that mobile users will follow the shortest path to the destination. Initially, this filter calculates the shortest path (p_{min}) from the current location to the destination for each mobile user and then selects some border points within a boundary distance d from the shortest path, as shown in Fig. 2(c). The distance d indicates the level of steadiness: if a mobile user follows the calculated shortest path, a small value of d is sufficient. The shortest path-based filter then stores the selected candidate border points with the calculated shortest path for each mobile user. When a mobile user m wakes up, this filter retrieves the stored candidate border points of m and then finds the nearest border point, among the retrieved border points, using the ROADALARM basic algorithm. Finally, this approach calculates the hibernation time using the nearest border point in the same way as is done in our basic algorithm.

The shortest path-based filter has a built-in resilience to handle mobile clients which go out of the reduced search space based on the shortest path as show in Fig. 2(d). When a mobile user m wakes up, this filter calculates the distance from the stored shortest path of m. If the distance is larger than d, it recalculates the search space based on the client's current shortest path to the destination.



(a) Alarm success rate (b) Average Hibernation time

Fig. 3. Comparison of segment length and travel time based approaches

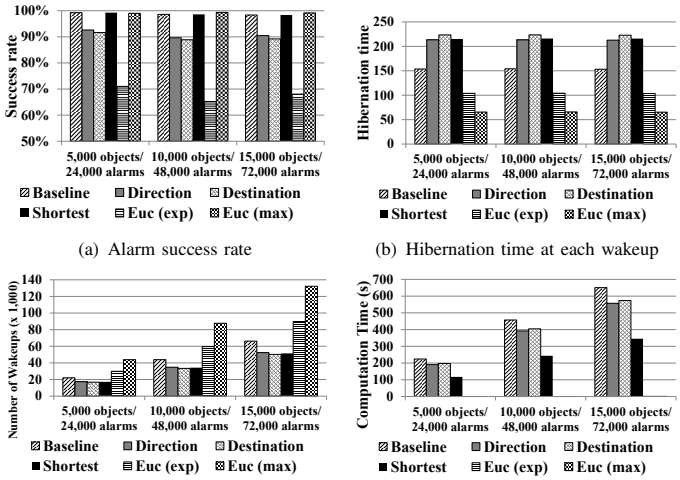
V. EXPERIMENTAL EVALUATION

For our experiments, we use *gt-mobisim* simulator [8] to generate mobility traces on real road networks downloaded from U.S. Geological Survey (USGS [9]). The mobility traces are generated on a map of northwest Atlanta, which covers about 11 km (6.8 miles) by 14 km (8.7 miles), using the random trip model [12]. The road network consists of four different road types: residential roads and freeway interchange with 30 mph speed limit (48 km/h), highway with 55 mph limit (89 km/h) and freeway with 70 mph limit (113 km/h). Ranges of spatial alarms are chosen from a Gaussian distribution with a mean of 50 m, and standard deviation of 10 m. This setting generates spatial alarms with a wide spectrum of sizes of target regions. We use 50 m as the boundary distance d of the shortest path-based filter. We will refer to the ROADALARM basic approach as "*baseline*" for short.

A. Comparison with existing methods

We first compare segment length-based approaches and travel time-based approaches as shown in Fig. 3. These experiments use 15,000 mobile users (and about 72,000 spatial alarms) and 180° as the Θ value of the current direction-based and destination-based motion-aware filters. Each user has different number of spatial alarms, given by Zipf distribution with five alarms as the most common value (i.e. rank 1). We exclude the results of network expansion-based methods since they cannot scale to 15,000 mobile users. The alarm success rate for travel time-based approaches is higher than the corresponding segment length-based approaches as shown in Fig. 3(a). This is primarily because segment length-based approaches can miss some alarms if mobile users follow paths having shorter travel time. On the other hand, the average hibernation time of each travel time-based approach is shorter since the travel time on the segment length-based shortest path is always equal to or longer than that on the travel time-based shortest path for same source and destination location. Without loss of generality, in the rest of the experiments, we include the results of only travel time-based approaches for simplicity.

The first set of experiments compares our approaches with existing Euclidean space-based methods in Fig. 4. We use a mobile user population with size ranging from 5,000 to 15,000 and each user has different number of alarms, given by Zipf distribution with five alarms as the most common value. The total number of alarms ranges from about 24,000 to 72,000. The success rate is shown in Fig. 4(a). The shortest path-based filter has almost the same success rate as the Euclidean distance-based approach using the global maximum speed and



(a) Alarm success rate (b) Hibernation time at each wakeup

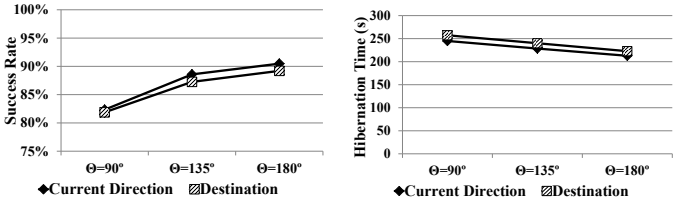
(c) Number of Wakeups (d) Total Computation Time

Fig. 4. Comparison with existing Euclidean space-based approaches

the *baseline*. The Euclidean distance-based approach using the expected speed has the lowest success rate for two reasons. First, it fails to consider spatial network constraints that mobile users need to obey. Second, the expected speed is estimated based on the maximum speed limit and the past speed and it may not accurately reflect the current speed in reality.

Fig. 4(b) shows the average hibernation time of mobile users. The longer the hibernation time is, the more energy the mobile clients can conserve. The hibernation time of the shortest path-based filter is three times longer than that of the Euclidean distance-based approach using the global maximum speed and 40% longer than that of the *baseline*. This result also shows that mobile users of the shortest path-based filter can conserve much more energy while ensuring high success rate. The Euclidean distance-based approach using the global maximum speed has the shortest hibernation time since it pessimistically utilizes the Euclidean distance and the global maximum speed to calculate the hibernation time. The *baseline* has shorter hibernation time than the shortest path-based filter since it always selects a border point having the shortest travel time without considering motion characteristics of mobile users. Furthermore, the destination-based filter has a little longer hibernation time than the shortest path-based filter at the cost of high success rate.

Fig. 4(c) experimentally proves that the number of wakeups is inversely related to the hibernation time. The smaller number of wakeups also indicates the lower server loads since the server computes the hibernation time whenever a mobile user wakes up. Fig. 4(d) shows the total computation time to calculate the hibernation time. The shortest path-based filter has 45% faster computation time than the *baseline*. The current direction-based and destination-based filters have better computation time than the *baseline*, but worse than the shortest path-based filter. Fig. 4 shows that the Euclidean distance-based approaches have low hibernation time and lower success rate and high number of wakeups at client, even though its computation time is negligible.



(a) Alarm success rate (b) Hibernation time

Fig. 5. Effect of the steady motion degree Θ

B. Effect of the steady motion degree

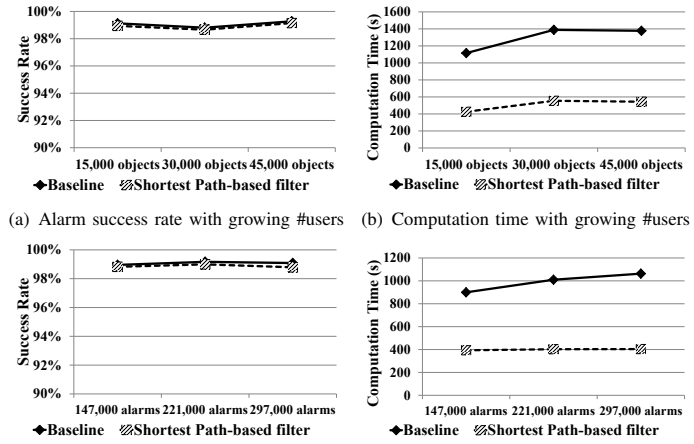
We investigate the effect of different settings of the steady motion degree Θ on success rate and hibernation time with 15,000 mobile users and about 72,000 spatial alarms as shown in Fig. 5. The success rate for the current direction-based and destination-based filters increases as Θ values increase, because more border points are selected as shown in Fig. 5(a). Fig. 5(b) shows that the average hibernation time decreases with growing Θ values. This is because border points having shorter travel time are newly selected to calculate the hibernation time as the search space defined by Θ increases.

C. Effect of growing number of users and alarms

Fig. 6(a) and Fig. 6(b) evaluate the scalability of our approaches by increasing the number of mobile users. Total 300,000 spatial alarms are deployed for this set of experiments and the number of mobile users increases from 15,000 to 45,000. Each user has zero to 30 spatial alarms, given by Zipf distribution with 15 alarms as the most common value, and all spatial alarms are private. We think this setting deploying 45,000 mobile users is realistic on this road network of northwest Atlanta, in which the total length of all road segments is 1384 km (865 miles), since there is a mobile user every 31 m (102 feet) on average. We include the measurement results of only the *baseline* and the shortest path-based filter as they have high success rate compared to other methods. Fig. 6(a) confirms that our approaches ensure the high success rate with growing number of mobile users. In terms of the total computation time, there is no increase from 30,000 to 45,000 users since with fixed alarms, many users have no spatial alarms as shown in Fig. 6(b).

Fig. 6(c) and Fig. 6(d) show the scalability of our approaches by increasing the number of spatial alarms with 15,000 mobile users. We increase the most common value of Zipf distribution from 10 to 20 and thus the total number of alarms grows from about 147,000 to 297,000. Fig. 6(c) verifies that our approaches ensure the high success rate with an increasing number of spatial alarms. Fig. 6(d) shows that the computation time of the shortest path-based filter increases only slightly with the growing number of spatial alarms. This is primarily because the shortest path-based filter selects only border points having high probability to be hit and thus the increased number of spatial alarms has no huge impact on the selected border points by the shortest path-based filter.

In summary, our experimental results show that the shortest path-based filter outperforms the rest in most cases since this approach ensures high success rate while reducing the



(a) Alarm success rate with growing #users (b) Computation time with growing #users

(c) Alarm success rate with growing #alarms (d) Computation time with growing #alarms

Fig. 6. Effect of growing number of users and alarms

computation cost of servers and conserving energy of mobile clients. For those applications in which high success rate is required, both the *baseline* and the the shortest path-based filter are good options. Especially, for some applications in which the battery power of mobile clients is not a serious problem, the *baseline* may be a better choice since it has a slightly higher success rate than the shortest path-based filter.

VI. Related Work

Existing research on spatial alarms and location reminders mainly focuses on the Euclidean space. [3] proposes an approach to process spatial alarms in the Euclidean space by combining spatial indexes such as R-tree and Voronoi Diagram with the safe period. [2], [5] develop a safe region-based approach for spatial alarm processing in the Euclidean space.

We would like to note that spatial alarms are fundamentally different from continuous spatial queries in terms of their objectives and target applications. Continuous spatial queries, such as finding the restaurants 2 miles around me, are defined based on the current location of mobile users for finding points of interest within a predefined range from the current location of a mobile user. [7], [14] use Euclidean distance while [10], [13] use road network distance in continuous spatial query modeling and processing. Continuous spatial queries are inadequate and incur poor performance for location trigger-based applications, such as hazard alert systems and location-based advertisement. In contrast, spatial alarms are independent of the current location of mobile users and are defined based on some future location of interest, such as "alert me when I am 2 miles to the public library in Buckhead". Clearly, (1) the focal of a spatial alarm is not the current location of the mobile client but the current location of the alarm target (e.g., the public library in Buckhead), and (2) the spatial alarm evaluation should not be triggered until the mobile client who subscribed to the alarm is in the vicinity of the alarm target. Thus, spatial alarms are essential building blocks for location trigger-based applications, such as location-based advertisement applications.

VII. Conclusion

We have presented RoadAlarm – an efficient and scalable service architecture and a suite of algorithms for processing road network-based spatial alarms. By utilizing spatial constraints on road networks and mobility patterns of mobile users on spatially constrained road networks, we have shown through extensive experiments that the RoadAlarm service architecture can significantly reduce the computation time for calculating hibernation time upon wakeups of mobile users, compared to the state of the art conventional approaches.

This paper has made three technical contributions. First, we present the RoadAlarm service architecture that defines road network-based spatial alarms as star-shaped subgraphs and uses the border points and road network distance to represent the boundary of road network-based spatial alarms. By making the spatial alarm service architecture road network aware, we are able to capitalize on the spatial constraints of mobile users' movements to improve the performance and scalability of the RoadAlarm service engine and the energy efficiency of RoadAlarm client on mobile devices. Second, we present the RoadAlarm basic algorithm that combines subscription filter with Euclidean Lower Bound (ELB) filter to reduce the number of unnecessary shortest path computations. Third but not the least, we introduce three motion-aware filters to further reduce the computation cost by minimizing the number of unnecessary shortest path computations as well as client wakeups by making use of the steady motion-based motion patterns of mobile users traveling on a road network.

Acknowledgment

This work is partially supported by grants from NSF NetSE, NSF CyberTrust, an IBM faculty award and a grant from Intel ISTC on Cloud Computing.

References

[1] Apple. Optimize your settings. http://www.apple.com/batteries/iphone.html.
[2] B. Bamba, L. Liu, A. Iyengar, and P. S. Yu. Distributed processing of spatial alarms: A safe region-based approach. In *ICDCS*, 2009.
[3] B. Bamba, L. Liu, P. S. Yu, G. Zhang, and M. Doo. Scalable processing of spatial alarms. In *HiPC*, 2008.
[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
[5] M. Doo, L. Liu, N. Narasimhan, and V. Vasudevan. Efficient indexing structure for scalable processing of spatial alarms. In *GIS*, 2010.
[6] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24, June 1999.
[7] H. Hu, J. Xu, and D. L. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *SIGMOD*, 2005.
[8] gt-mobisim. http://code.google.com/p/gt-mobisim/.
[9] U.S. Geological Survey. http://www.usgs.gov/.
[10] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, 2006.
[11] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, 2003.
[12] P. Pesti, L. Liu, B. Bamba, A. Iyengar, and M. Weber. Roadtrack: scaling location updates for mobile clients on road networks with query awareness. *Proc. VLDB Endow.*, 3, 2010.
[13] S. Xing, C. Shahabi, and B. Pan. Continuous monitoring of nearest neighbors on land surface. *Proc. VLDB Endow.*, 2, August 2009.
[14] X. Yu, K. Q. Pu, and N. Koudas. Monitoring k-nearest neighbor queries over moving objects. In *ICDE*, 2005.