mTrigger: An Event-based Framework for Location-based Mobile Triggers

Ling Liu
Georgia Institute of Technology, USA
Bhuvan Bamba
Georgia Institute of Technology, USA
Myungcheol Doo
Georgia Institute of Technology, USA
Peter Pesti
Georgia Institute of Technology, USA
Matt Weber
Georgia Institute of Technology, USA

ABSTRACT

Location-based triggers are the fundamental capability for supporting location-based advertisements, location-based entertainment applications, personal reminders, as well as presence-based information sharing applications. In this chapter, we describe the design and the implementation of mTrigger, an event-based framework for scalable processing of location-based mobile triggers (location triggers for short). A location trigger is a standing spatial trigger specified with the spatial region over which the trigger is set, the actions to be taken when the trigger conditions are met, and the list of recipients to whom the notification will be sent upon the firing of the location trigger. The mTrigger framework consists of three alternative architectures for supporting location triggers: (1) the client-server architecture, which allows mobile clients to register and install location triggers of interest on the mTrigger server system; the server being responsible for processing location triggers, performing associated actions and sending out notifications upon firing of triggers; (2) the client-centric architecture, which enables mobile users to manage and process location triggers on their own mobile clients; and (3) the decentralized peer-to-peer architecture, which allows mobile users to collaborate with one another in terms of location trigger processing. The server-centric architecture is particularly suitable for supporting public and shared location triggers, enabling effective sharing of location trigger processing among multiple users. The client-centric architecture is more suitable for users possessing mobile clients with high computational capacity and more sensitive to the location privacy of their location triggers. The decentralized peer-to-peer architecture provides on-demand and opportunistic collaboration in terms of location trigger evaluation. Clearly, the performance optimizations for server-centric architecture should focus on efficient and scalable processing of location triggers by reducing the bandwidth consumption and the amount of

redundant computation at the server; whereas, the performance optimizations for client-centric architecture and decentralized architecture should also take into account energy efficiency of mobile clients in addition to computational efficiency. In addition, processing of location triggers with moving target of interest requires the knowledge of position information of the moving target and may not be suitable for the client-centric architecture. This chapter will describe the design principles and the performance optimization techniques of the mTrigger framework, including a suite of energy-efficient spatial trigger grouping techniques for optimizing both wake-up times and check times of location trigger evaluations.

Keywords*: location-based services, location trigger, mTrigger, client-server architecture, location trigger manager, optimization module, safe period, safe region, bitmap encoded safe region*

1. Introduction

Location-based services such as location-based advertisement, location-based entertainment and location-based personal assistant are emerging business applications that demand location-based mobile triggers. Location triggers are standing spatial triggers. Similar to time-based triggers that are used to remind us of the arrival of a future reference time point, location triggers are set on a spatial location of interest, which subscribers of the trigger will travel to at some future time instant. Companies or merchants may use location triggers to support location-based advertisements; for example, Bloomingdale's may send a 20% sale coupon to its medallion member shoppers who are located within five miles of its stores. Individuals use location-based triggers to set up personal reminders indicating the arrival to a spatial location of interest. For instance, a user could set a spatial alarm (Spatial Alarms Project, n.d.) on her mobile client, which alerts her whenever she is near the dry cleaner or her favorite grocery store in her neighborhood, reminding her to pick up or drop her dry cleaning items, or automatically retrieving her stored grocery list.

Dey and Abowd (Dey, A., & Abowd. G., 2000) describe a context-aware system for supporting reminders in order to provide appropriate signals at appropriate times. For example, a reminder for bringing a paper for a meeting is most effective when the user is leaving her office to head for the meeting room. (Dey, A., & Abowd. G., 2000) primarily deals with building a context-aware toolkit for supporting reminder delivery at appropriate times. The ability to locate users using GPS, cell phone positioning and other navigational systems makes context-aware reminder systems feasible. *PlaceMail* (Ludford, P., Frankowski, D., Reily, K., Wilms, K., & Terveen, L., 2006). studies issues related to location-based information systems in order to support useful location-based reminder systems and functional place-based lists. The study determines that effective reminder delivery depends on people's movement patterns through an area and the geographic layout of the space. However, none of the previous work emphasizes the ability to process reminders (or more generally, location-based triggers) *efficiently* from a systems-based perspective. This chapter focuses on optimization techniques which should be deployed for efficient and scalable processing of location-based triggers in different systems-based architectural settings, including a server-centric, client-centric and decentralized architecture. We also consider bandwidth and energy constraints on the client side, which are resource-

constrained devices despite significant enhancements in the past few years, for efficient trigger evaluation. More concretely, we develop safe period and safe region-based optimizations in order to facilitate efficient and scalable processing of location-based triggers.

A mobile location-based trigger (location trigger for short) is defined as a standing spatial trigger with four mandatory components: (1) the spatial region over which the trigger is set, (2) the list of subscribers or recipients to which the notification will be sent when the trigger condition is met, (3) the actions to be taken upon firing of the trigger, and (4) the stop condition which specifies the termination constraint for the trigger. Mobile users can define and register their location triggers with the mTrigger (mTrigger, n.d.) system. Once a location trigger is installed, the system will start monitoring the spatial region of the trigger and whenever any mobile subscribers of the trigger enter the spatial region, the location trigger is fired and the specified actions are executed. The result of the action will be sent to the subscriber (recipient) of the location trigger. For example, "*inform me whenever I am within two miles of the dry cleaner at the crossing of Druid Hill Road and Briarcliff Road in the next four weeks*" is a location trigger installed on the spatial region two miles around the dry cleaning store at the crossing of North Druid Hill and Briarcliff with a stop condition of four weeks. The action is simply a notification sent to the owner of this location trigger.

Location triggers are classified based on the scope of their recipients and the mobility characteristics of their monitoring targets and their subscribers. We define three classes of location triggers based on the scope of the recipients: Private, Public and Shared. A location trigger is private if only the owner of the trigger is its recipient. In contrast, public location triggers are those triggers that are installed by their respective owners and are shared by all users of the system. Notifications of public triggers are sent to all users who have subscribed to the triggers and are online at the time of notification. Traffic notifications, or notifications related to hazardous road conditions are typical examples of public location triggers. While defining a location trigger, the owner may specify a list of people with whom she wants to share the trigger. We call this class of location triggers shared triggers. In this case, only people who are specified on the recipient list and confirm as subscribers of the trigger will receive a notification whenever the location trigger is fired.

Location triggers are also categorized based on the mobility characteristics of their subscribers and monitoring targets. Three types of location triggers are considered in the design of mTrigger: (Mobile *Subscriber*, Static *Monitoring Target*), (Mobile *Subscriber*, Mobile *Monitoring Target*), and (Static *Subscriber*, Mobile *Monitoring Target*). A location trigger is of the type (Mobile *Subscriber,* Static *Monitoring Target*) if it has a static monitoring target, such as the dry cleaning store in the previous example, and a moving subscriber. A location trigger is of the type (Mobile *Subscriber*, Mobile *Monitoring Target*) if its monitoring target, as well as its subscriber are both on the move. For example, "*notify me when all my friends are within a five mile vicinity on highway I-85 North*" is a typical example of this type of trigger. A location trigger is said to be of the type (Static *Subscriber,* Mobile *Monitoring Target*) if its monitoring target is moving but its subscriber is still. In this case, the spatial region of the location trigger moves as the monitoring target moves, but the position of the subscriber remains unchanged. "*Notify me whenever bus No. 5 is five minutes away from the bus-stop near my office*" is an example of a Static *Subscriber* Mobile *Target Trigger*.

In this chapter, we develop an event-based framework, called mTrigger, for scalable processing of location-based mobile triggers. The mTrigger framework consists of three alternative architectures for supporting location triggers:

- The *client-server architecture*, which allows mobile clients to register and install location triggers of interest on the mTrigger system server. The server is responsible for processing location triggers, performing associated actions and sending out notifications upon the firing of triggers. The server-centric architecture is particularly suitable for supporting public and shared location triggers, enabling effective sharing of location trigger processing among multiple users. Clearly, the performance optimizations for the server-centric architecture should focus on efficient and scalable processing of location triggers by reducing the bandwidth consumption and the amount of redundant computation at the server (Bamba, B., Liu, L., Yu, P. S., Zhang, G., & Doo, M., 2008), (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009).
- The *client-centric architecture*, which enables mobile users to manage and process location triggers on their own mobile clients. The client-centric architecture is particularly suitable for supporting private location triggers and for mobile users who are more sensitive to the location privacy (Bamba, B., Liu, L., Pesti, P., & Wang, T., 2008), (Gedik, B., & Liu, L., 2008) of their location triggers. The performance optimization for client-centric architecture should take into account energy efficiency of mobile clients in addition to computational efficiency (Murugappan, A., & Liu, L., 2008).
- The *decentralized peer-to-peer architecture* is built on top of a client-centric architecture, and is suitable for processing location triggers shared among a group of mobile users. The decentralized architecture relies on collaboration among mobile users in close vicinity and with common interests, to share location trigger processing costs.

In addition, processing of location triggers with moving target of interest requires the knowledge of positioning information of the moving target and thus is not suitable for the client-centric architecture.

In this chapter, we focus on the event-based framework of mTrigger in terms of both client-server architecture (Section 2) and client-centric architecture (Section 4) and the set of server-side optimization techniques for scalable processing of location triggers (Section 3). We evaluate the mTrigger optimization techniques in terms of accuracy and scalability (Section 6), and conclude with the summary and a discussion of future work.

2. **mTrigger Client-Server Architecture**

The server-centric architecture consists of five main components.

- The first component is the mobile client module, which handles the interaction of mTrigger Engine with its mobile clients as well as the communication with the mTrigger application server.

- The second component is the mTrigger application server, which supports a variety of application plug-ins that enable the mTrigger engine to support different location services such as spatial alarms, mGraffiti, location-based games, location-based advertisements, and so forth. Each application plug-in typically implements the transformation of application specific service requests to location trigger expressions, which in turn registers with the mTrigger engine through the Trigger Manager.
- The third component is the location trigger manager which handles the trigger registration, mobile user registration, user account management and mTrigger administrative assistance functions. All location triggers and user information are maintained in the mTrigger database.
- The fourth component is the optimization module, which offers a suite of server-side optimization techniques, such as spatial trigger indexing techniques, safe period and safe region techniques for optimizing both wake-up times and check times of location triggers shared by multiple subscribers.
- The fifth component is the location trigger processing module, which evaluates location triggers closer to the mobile subscribers of interest based on the safe region or safe period prediction techniques. Concretely, a mobile client will wake up whenever she moves outside the safe region of one of her location triggers or whenever the safe period for one of her location triggers has expired. Furthermore, a location trigger will be checked through its trigger condition evaluation whenever a mobile user enters the corresponding location trigger monitoring region.

Fig. 1 shows an overview of the client-server architecture of the mTrigger system. In this architecture, only a thin client of mTrigger is required to run on the mobile clients, which communicates with the mTrigger engine through the mTrigger application plug-in services.

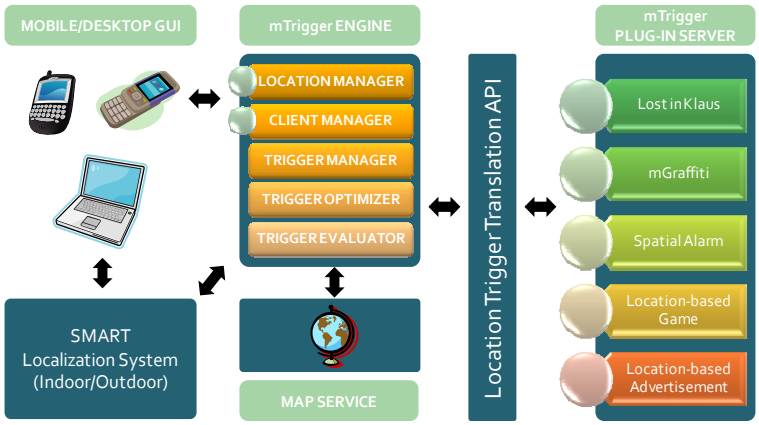


Figure 1. The mTrigger Client-Server Architecture

Positioning: We assume that each mobile device either has a GPS unit or is equipped with Wi-Fi capability. The GPS unit provides longitude, latitude, and altitude of current location. Given the poor performance of GPS indoors, we use Wireless Positioning Systems, such as SMART, to address indoor positioning problems. SMART uses area localization techniques

to provide an approximate location of a mobile user with a Wi-Fi enabled device, within an area (for example, the Georgia Tech campus). One of the positioning techniques used in SMART is the Wi-Fi Access Points-based location fingerprinting technique. There are typically two approaches for location acquisition. One approach is to require all mobile users interested in receiving location-based services to update their current position periodically or use one of the inference-based location update techniques, such as dead reckoning. The second location acquisition technique is to apply location determination methods such as triangulation, trilateration, and so forth.

***Mobile Device**:* A thin mTrigger client will be required for all users in the mTrigger client-server architecture. In the mTrigger thin client, we allow users to install a location trigger through both text interface and map-enabled GUI. By enabling a map-based interface, the mobile user can simply mark the spatial area of interest on the map to define the monitoring region of a location trigger. An example usage scenario of such a thin client is shown in Figure 2.

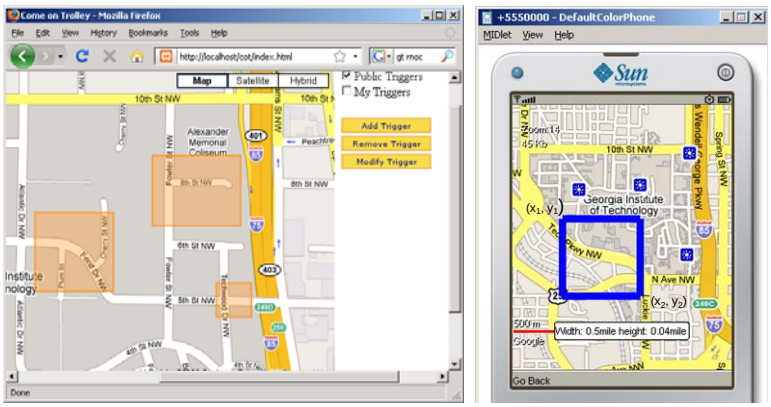


Figure 2. Bounding Box for a Trigger Region and Installed Triggers

Location Triggers: A mobile user can install many location triggers that are shared among a group of mobile users. On the other hand, public location triggers can be installed by authorized users only. The mTrigger (mTrigger, n.d.) location trigger evaluation engine consists of the client manager, trigger manager, wakeup and trigger evaluation optimization module and location trigger processing module. Each mobile user can register her location triggers directly with the mTrigger engine or through the mTrigger application server plug-in services. As mentioned earlier, each plug-in handles the transformation of the application specific location service request, such as a spatial alarm, or a location-based advertisement request to a semantically equivalent location trigger expression, which in turn is submitted to the mTrigger system for registration, processing and notification.

One of the design goals of the mTrigger event-based framework is to provide middleware support for many location-based services and applications that use location triggers as a fundamental capability. This design framework enables applications to provide the proper mTrigger plug-in services. Typically, different applications may use different compositions of location trigger elements and specification. Some use all the elements of location triggers and others use only some of the elements. Typically the service request format converter may add or remove some elements, or overcome the mismatch between the mTrigger engine and plug-in server. A database is used to provide persistent storage for mobile users along with

their positioning information and corresponding safe regions as well as the location triggers registered with the mTrigger system.

Formally, a location trigger *t* is specified in terms of eight elements:

t = (*tID*, *Owner*, *Region*, *Event, Subscriber*, *ActionType, Action, Start, Stop*).

- *tID* is a unique identifier that is assigned to each location trigger upon its registration.
- *Owner* is specified by the user id of the owner/creator of this location trigger.
- *Region* is specified by a landmark and a spatial monitoring area nearby or around the landmark, on which the location trigger is set. The monitoring area of the region could be represented by a rectangular bounding box. For simplicity, in the rest of this chapter, we represent each of the trigger regions by a rectangular bounding box, denoted by (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) represent the bottom-left and top-right vertices of the bounding box, as shown in Fig. 2.
- *Event* specifies non-spatial event-based trigger conditions associated with the installed trigger, represented by <*monitored attribute, condition predicate*>. For example the price of gasoline at a gas station is one such non-spatial condition.
- *Subscriber* specifies a list of mobile subscribers to which the trigger notifications should be delivered.
- *ActionType* defines the type of action to be performed when the location trigger is evaluated to be true. *ActionType* is specified by a three element vector <*format, file name, application name*>: (1) *format* specifies the multimedia data format, such as text, URL, image, audio or video; (2) *file name* is the multimedia data file to be sent along with the notification; and (3) *application name* specifies the application plug-in name, such as mGraffiti, Spatial Alarm, or Location-based advertisement.
- For each type of action, the *Action* field contains the delivery and display method, e.g., text or other multimedia message, or an executable method.
- *Start* defines the condition when the location trigger monitoring should start. A simple format for *Start* is a time point, such as 10AM on October 20, 2008.
- *Stop* defines a termination condition for the location trigger. It can be specified as either a time duration, such as 30 minutes (half an hour) from the installation time, or a time point, such as 8AM on October 20, 2008 (two weeks from the installation).

Recall the example location trigger, "*Notify me whenever I am within 2 miles of the dry cleaner at the corner of Druid Hill Road and Briarcliff Road in the next four weeks*". The trigger region is defined by the landmark object - the dry cleaner at the corner of Druid Hill and Briarcliff, and the spatial monitoring region (two miles) around the given landmark object. The subscriber and the owner are the same for this location trigger. The type of the action in this case is simply a notification message sent to the subscriber with Spatial Alarm as the plug-in name. To make the notification more user-friendly, the mTrigger system allows the users to supply photo images or symbols in addition to the text message they wish to receive when the trigger condition is met. In the context of this example, the owner of the location trigger may provide the dry cleaning shop name and a recent photo of the store in JPEG format at the time of trigger installation. The action, by default, will be a method call

for displaying the store name and the photo image on the screen of the mobile client. When there is an absence of user-supplied notification information, the text description of the location trigger will be sent to the subscriber.

Location Trigger Processing Engine: Figure 3 provides an overview of the mTrigger engine design. The mTrigger engine manages the mobile users' accounts and their service requests for installation, removal and modification of location triggers through the Client Manager and the Location Trigger Manager. The *Client Manager* handles all communication between mobile devices and the mTrigger Engine. We provide two ways to interface with mobile clients: one is to allow mobile clients to directly request registration, location update and trigger installation. Another is to allow mobile clients to communicate with the mTrigger system indirectly through the application specific plug-in service. In addition, the Client Manager manages the user account for all registered users, through account activation/deactivation functions. User accounts may include the notification destination address of the user to which the notification of her location triggers will be delivered. Users

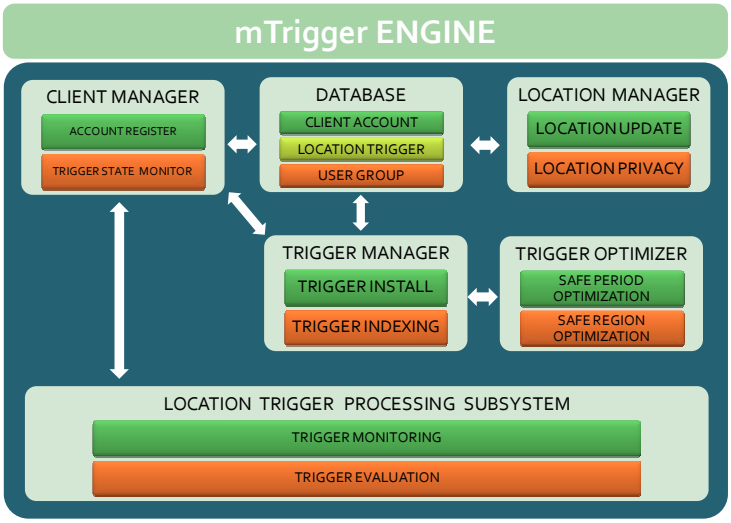


Figure 3. mTrigger Engine Architecture

can revise the notification recipients and the notification address for each location trigger installed, as some triggers are shared by a number of users. The client manger will also handle the group creation and maintenance. We use the mutual agreement protocol to establish the sharing of location triggers among multiple users in order to alleviate any location privacy concerns. For example, assume Alice wants to send a greeting video clip to Bob when Bob gets back home from a business trip. In order to be informed about Bob's arrival by the mTrigger system, Alice needs to obtain Bob's agreement that allows Alice to install a location trigger around the neighborhood of Bob's home. We plan to incorporate stronger location privacy protection (Bamba, B., Liu, L., Pesti, P., & Wang, T., 2008), (Gedik, B., & Liu, L., 2008) in our second release of the mTrigger system. The Location Trigger Manager consists of the Trigger Installation module, Trigger Indexing module and Trigger State Management. The Location Trigger State Management module tracks the state of location triggers and serves requests on active triggers, terminated triggers and number of evaluations performed for each location trigger. The third component of the mTrigger engine is the *Position Manager*, which consists of two key components: Position Update Module and Location Privacy Module. The Position Manager handles the location tracking and

update functions for mobile clients according to their location privacy constraints. Two alternative types of location acquisition methods are supported in the mTrigger system: location determination through the SMART localization system, or a mobile client reporting model, where mobile users report their current location periodically or using a system supplied location derivation method to reduce the energy consumption and the server load due to frequent location updates. *Trigger Optimizer* is the fourth component of mTrigger and consists of the main optimization modules such as Safe Period Optimization and Safe Region Optimization (to be discussed in Section 3). Finally, the fifth component of the mTrigger engine is the *Location Trigger Processing Module*, which consists of Trigger Monitoring and Trigger Evaluation. The mTrigger engine uses a database to maintain user account information, location trigger registration and state management, group creation and maintenance, and for position updates of mobile users.

In principle, upon receiving a location update, the system should evaluate the relevant location triggers. If a trigger condition is met upon a location update of a mobile client, then a trigger notification will be sent to the registered subscriber(s). Clearly, when a mobile user is far away from all her installed location triggers, her location updates would not cause any of her active location triggers to be true. In order to reduce the energy consumption and the amount of unnecessary location trigger evaluation, the location trigger optimizer provides various data structures such as R-Tree, BR-Tree and Grid index structure. A suite of safe period and safe region-based optimization techniques offer significant savings by reducing the number of unnecessary wakeups and checks to be performed at the mTrigger server. We will describe some of the optimization techniques in Section 3.

It is important to note that the location trigger expression is designed as an internal modelling construct for location triggers in the mTrigger framework. For different applications, one can use application specific plug-ins to make the mTrigger engine transparent to its end users. We now illustrate this abstraction through a number of example applications.

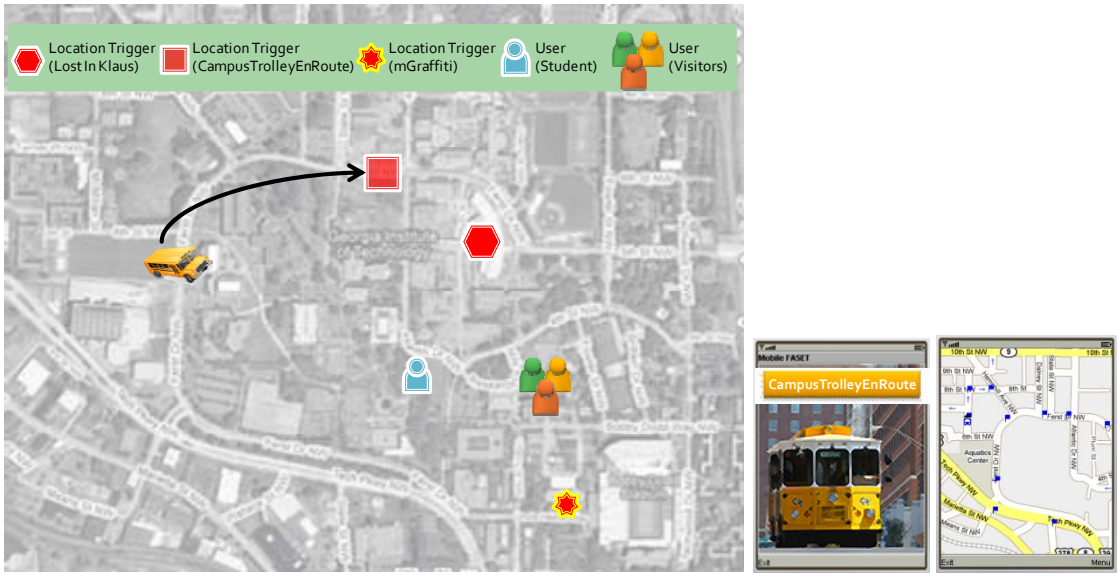


Figure 4. Location Triggers Example 1: CampusTrolleyEnRoute, campus shuttle tracking service offered on Georgia Tech Campus

CampusTrolleyEnRoute: The first example is a Georgia Tech Campus bus tracking application. Figure 4 depicts a map of Georgia Institute of Technology and the first prototype of the mTrigger system. Red rectangles represent previously installed location triggers. There are shuttles on the campus to transfer students and faculty from one building to another within the campus. The CampusTrolleyEnRoute application provides a valuable service in terms of providing the current bus location and an estimate of the time of arrival of the bus to the nearby bus stop. By building CampusTrolleyEnRoute using mTrigger, one can provide a much richer collection of services without additional development effort. For example, one can support spatial alarm type of services such as "*notify me when the yellow bus is five minutes away from the Klaus Advanced Computing Building (KACB)*" or "*when the yellow trolley is about half a mile away.*" One can also support services like "*when the trolley approaches the KACB bus stop, ring CERCS office director to notify that IBM visitors are arriving*". Figure 4 shows the current location of the Trolley buses being monitored.

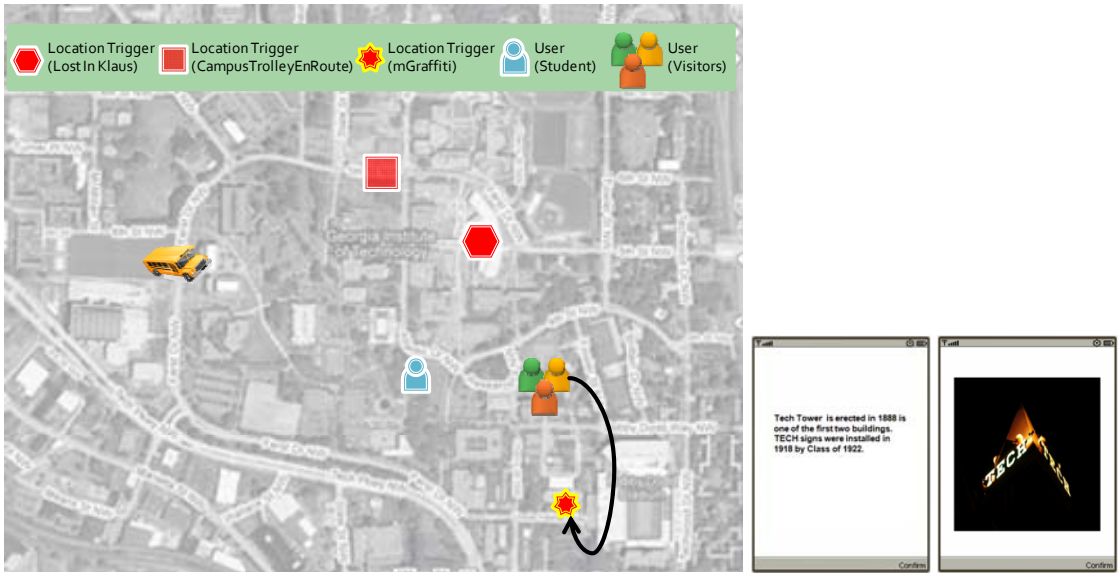


Figure 5. Location Trigger Example 2: mGraffiti – location-based virtual graffiti

mGraffiti: The second example application is called mGraffiti; a sample screenshot is shown in Figure 5. The main idea of mGraffiti (mGraffiti, n.d.). is to provide location-based virtual graffiti capability for PDA, Smartphone and Tablet PC users. With virtual graffiti, anyone can become a graffiti writer, including hikers, holiday-goers, and frequent travellers. In the context of the Georgia Tech campus, visitors can benefit from location-dependent virtual Graffiti to walk through the campus and visit the major buildings of the campus, such as the Tech Tower, stadium, recreation center, etc. Instead of writing down messages on walls or carving them on trees, visitors can express their impressions of the campus and leave graffiti or photos of the campus locations of interest using mGraffiti. Alternatively, we can also use Spatial Alarms to implement virtual graffiti as a public trigger or a shared location trigger with a simple action of displaying the text message and the multimedia attached with the text message. When visitors arrive at certain buildings or places of interest, the graffiti message and attached photos will pop up on the mobile client. Figure 5 shows an example of a textual graffiti message at Tech tower.

Spatial Alert: The third example application is Spatial Alert. Spatial Alerts are also called spatial reminders, used widely for reminding mobile users when they approach some location of interest, defined either by themselves in the form of spatial reminders, or defined by their friends through spatial alerts (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009). In fact, spatial alerts are a specific location trigger, in which notification is the only type of action being taken when the location trigger conditions are met. Examples of spatial alerts include "*Remind me when I am within two miles of the dry cleaning store at the crossing of Druid Hill Road and Briarcliff Road"* or "*Remind me when any of my friends are present in the Starbucks coffee store across the Ferst Drive from KACB"*. The main distinction between mGraffiti and the Spatial Alert service lies in the usage model and the context in which a location trigger is utilized.

Lost in Klaus: This is an indoor navigation system. Most new buildings on the Georgia Tech campus have complex building layouts and it is often difficult to navigate and find specific office locations (Lost in Klaus, n.d.). This is especially true for visitors, including students or staff members that do not have an office in the building. This project provides indoor shortest path-based navigation on mobile devices. The application utilizes Wi-Fi based localization to locate a mobile user in the building, and shortest path-based navigation to show how to get to a targeted office or meeting room through the shortest travel path. There are several ways one can employ location trigger capabilities to provide value added services. For instance, the KACB building manager can set a public location trigger on the entry points of the building. Upon entering the building, the Lost in Klaus client application will be launched on the client devices of mobile users. It determines the current position of a mobile user and provides the shortest path navigation to the destination office or classroom or meeting room. One can also extend this navigation system by allowing visitors or students to leave a note at a particular office in case the professor or the meeting organizer is absent at the time of the meeting. This can be easily supported by directly employing the mGraffiti plug-in service. Another desired capability is to monitor a specific room and notify the subscriber whenever the professor enters the room, or the door of the office being monitored is open. This can be provided through the Spatial Alarm plug-in to the mTrigger system.

3. Scalable Processing of Location Triggers: Server-side Optimizations

Processing of location triggers requires meeting two demanding objectives: *high accuracy*, which ensures no location triggers are missed, and *high scalability*, which guarantees that location trigger processing is highly efficient and can scale to a large number of triggers and the growing base of mobile users.

A naïve approach to location trigger processing is periodic evaluation at a high frequency. Concretely, each location trigger is evaluated periodically by testing whether the subscriber has entered the spatial region of the location trigger and whether the non-spatial trigger conditions are also met. High frequency in periodic evaluation is essential to ensure that none of the location triggers are missed. However, a proper setting of such a period is a hard problem. If the period is set too large, mobile clients may pass trigger monitoring regions without firing the location triggers in time, incurring a higher miss rate. On the other hand, if the update period is kept too small, it would make the mobile client wake up too frequently,

which is energy inefficient for mobile clients and unnecessarily increases the processing load at the server. As a result, the periodic evaluation approach to processing location triggers suffers from a number of drawbacks.

- First, the miss rate of location trigger evaluation is unpredictable as there is no appropriate technique for the system to determine the ideal trigger evaluation period. In the case of a high alarm miss rate, the system fails to meet the desired high accuracy requirement of location trigger processing.
- Second, the periodic trigger evaluation approach is expensive as it performs a large number of unnecessary evaluations at high frequency; hence, it is not scalable in the presence of a large number of location triggers and a large number of mobile users. In fact, the number of unnecessary evaluations increases as mobile users move farther away from the monitoring regions of their location triggers.

Consider an example scenario where a mobile user is moving in an area (say Atlanta) that is far away from her registered location triggers (say Miami). Obviously, it is unnecessary to evaluate her location triggers upon her location updates until she has traveled to a location that is near the city border of Miami. Motivated by this observation and the drawbacks of the periodic trigger evaluation, we argue that in a client-server platform, we need to optimize the location trigger evaluation by reducing the number of wakeups that the mobile clients need to perform for the purpose of location trigger evaluation. We also need to minimize the amount of unnecessary trigger processing the server needs to perform upon each location update of a mobile client. We now outline two optimization techniques used in the mTrigger system to enhance scalability of the location trigger processing engine while maintaining desirable accuracy: *safe period optimizations* and *safe region optimizations.*

3.1 Safe Period Optimizations

Safe period is defined as the duration of time for which it is safe not to check a given location trigger for a given mobile subscriber, as the probability that the location trigger condition is met for the subscriber is zero. Consider a subscriber S_i and a location trigger T_j. The safe period of trigger T_j with respect to subscriber S_i, denoted by $sp(S_i, T_j)$ can be computed based on the distance between the current position of S_i and the trigger monitoring region R_j, taking into account the motion characteristics of S_i and the monitoring target of location trigger T_j. Concretely, the two factors that influence the computation of safe period $sp(S_i, T_j)$ are (i) the velocity-based motion characteristic of the subscriber S_i, and (ii) the distance from the current position of subscriber S_i to the spatial region R_j of location trigger T_j. Thus the safe period $sp(S_i, T_j)$ can be computed as follows:

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V(S_i))}$$

Clearly, the distance measure between the current location of the mobile subscriber and the trigger region R_j is the first important parameter for safe period computation. The second important parameter is the velocity measure of the mobile subscribers or the mobile targets of the location trigger. *Euclidean distance* and *road network distance* are among the most commonly used distance measures. In order to ensure 100% location trigger accuracy, we use

maximum velocity of the mobile clients for the velocity measurement even though it provides a pessimistic estimate for the safe period. Other measures like *expected speed* may be used with precaution. We refer interested readers to (Bamba, B., Liu, L., Yu, P. S., Zhang, G., & Doo, M, 2008) for a more detailed discussion on these topics.

Concretely, the motion-aware safe period approach optimizes the location trigger processing through two basic mechanisms. First, we introduce the concept of safe period to minimize the number of unnecessary trigger evaluations, increasing the throughput and scalability of the system. We show that our safe period-based trigger evaluation techniques can significantly reduce the server load for location trigger processing, compared to the periodic evaluation approach, while preserving the accuracy and timeliness of location triggers. Second, we develop a suite of location trigger grouping techniques based on spatial locality of the triggers and motion behaviour of the mobile subscribers, which significantly reduces the safe period computation cost for location trigger evaluation at the server side. Only when the mobile client is entering a location trigger group region, say Group 3 in Figure 6, will all her location triggers within the group 3 be evaluated. By grouping similar location triggers based on their spatial locality and motion behaviour of the mobile users, we can, to some extent, reduce the number of trigger evaluations required.

Figure 6. Location Trigger Grouping

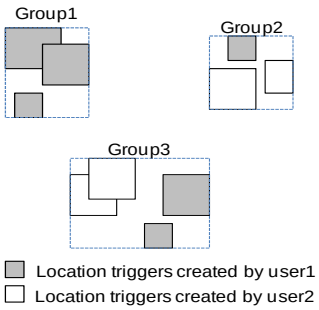
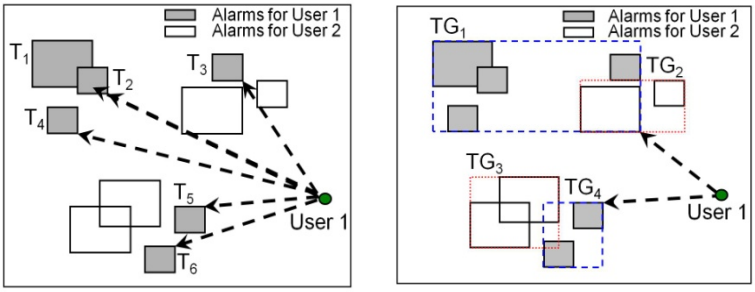


Figure 7. Safe Period-based Alarm Evaluation Techniques



(a) Per trigger safe period computation

(b) Subscriber specific trigger grouping-based safe period computation

Basic Safe Period Optimization: The safe period-based approach processes a location trigger in three stages. First, upon the installation of a location trigger, the safe period of the

trigger with respect to each authorized subscriber is calculated. Second, for each trigger-subscriber pair, trigger evaluation is invoked upon the expiration of the associated safe period, and a new safe period is computed. In the third stage, a decision is made regarding whether the location trigger should be checked or one should wait for the new safe period to expire. If the new safe period is smaller than a threshold t_δ, it means that the mobile client is entering the trigger monitoring region and the trigger condition is checked. Compared to periodic trigger evaluation, the safe period approach for location trigger processing reduces the amount of unnecessary evaluation steps, especially when the subscriber is far away from all her location triggers. On the other hand, the main cost of the basic safe period approach described in this section is due to the excessive amount of safe period computations.

Subscriber Specific Spatial Locality Grouping-based Safe Period Optimization: We present one such technique here which groups location triggers according to the subscriber-specificity at the first level, followed by the spatial locality of the alarms at the next level. Figure 7(a) displays the monitoring regions for a set of installed location triggers. The triggers for user 1 are marked by shaded trigger monitoring regions. Basic safe period evaluation computes the distance from each of the six triggers $\{T_i \mid 1 \leq i \leq 6\}$. Subscriber-specific spatial locality-based grouping performs a two level grouping: the first level grouping is on all subscribers and the second level grouping is on spatial alarms relevant to each subscriber. We use a B-Tree based implementation to speed up search on subscribers and an R-Tree implementation to capture spatial locality of location triggers for each subscriber in order to speed up the search of triggers. The underlying data structure is a hybrid structure which uses a B-tree for subscriber specific search at the first level and an R-tree for subscriber specific spatial alarm search at the second level. Figure 7(b) shows an example of this grouping. Triggers installed by user 1 are grouped together in TG_1 and TG_4 and may be fired only when the user is entering the MBRs of TG_1 or TG_4. Subscriber specific spatial locality-based grouping has two advantages over the basic safe period computation approach. First, the number of safe period computations is significantly reduced. Second, each alarm group contains alarms relevant to a single user, thus no irrelevant processing is performed. Our experimental results show that this approach is efficient in the presence of a large number of subscribers, and for a large number of private and shared alarms.

We evaluate the scalability and accuracy of our safe period-based optimization using a road network simulator. Our experimental results show that our motion-aware safe period-based approach offers significant performance enhancements, while maintaining high accuracy of location triggers, especially compared to the conventional periodic trigger evaluation approach.

3.2 Safe Region Optimizations

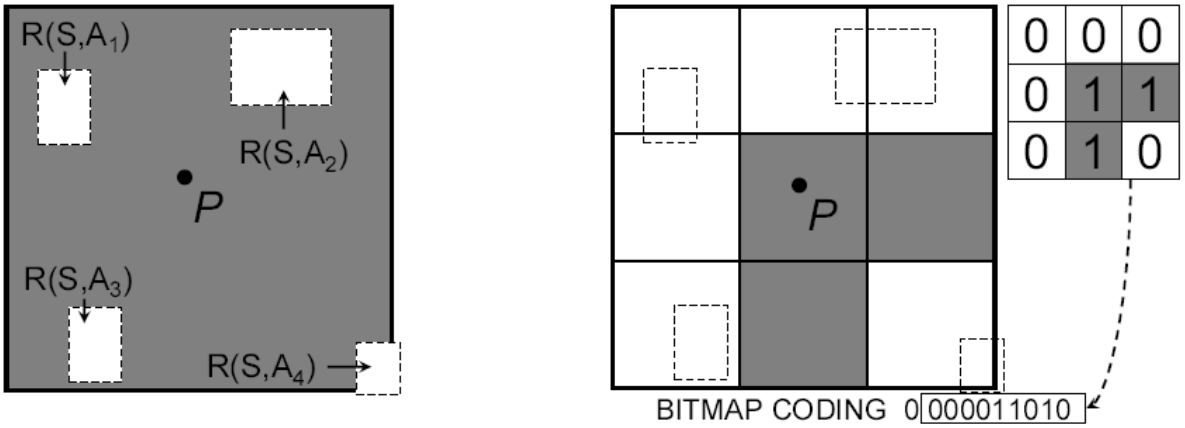
In contrast to safe period-based optimization, the safe region-based optimization computes a safe region for each subscriber, instead of each subscriber and trigger (or trigger group) pair. As long as the subscriber remains inside its computed safe region Ψ_S, the probability of any relevant triggers of this subscriber being triggered is zero. The server computes a safe region for each subscriber and communicates this safe region to the subscriber. The subscriber is responsible for monitoring its position within the safe region. Once the subscriber moves out of its safe region, the subscriber will provide a location update to the server which recomputes the safe region. A simple grid structure is overlaid on top of the *Universe of*

Discourse (or the geographical map of interest) while computing the safe region. By limiting the safe region computation to the current grid cell of the subscriber position, we can drastically reduce the safe region computation cost.

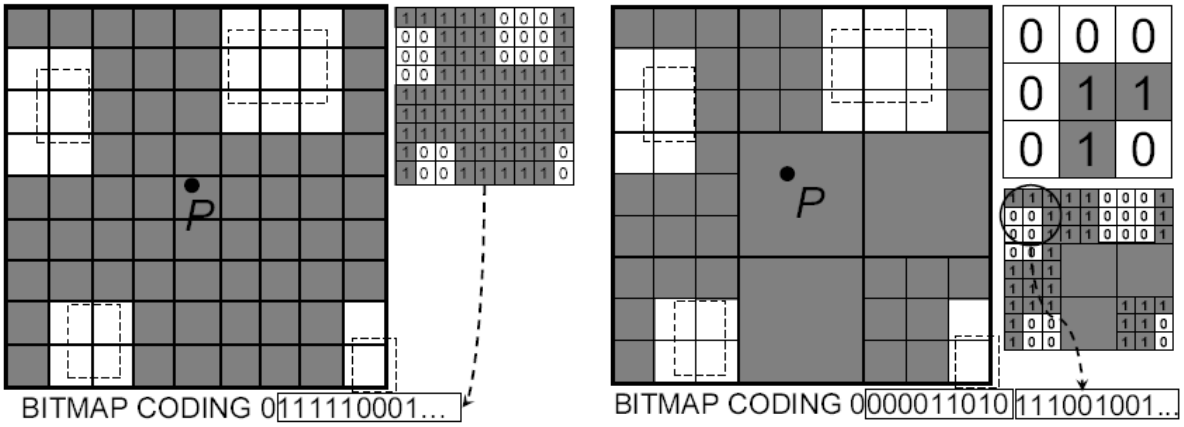
We have developed a number of techniques for safe region computation in (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009). In this section, we briefly review *Bitmap Encoded Safe Region (BSR)* techniques for safe region computation, which can express larger safe regions using a simple bitmap. We first describe a simple *Grid Bitmap Encoded Safe Region (GBSR)* computation technique and show that it does not provide an optimal way to compute safe regions efficiently and accurately. Then we develop an extension to the *GBSR* approach using a *pyramid* (Samet, H., 1990) data structure, referred to as the *Pyramid Bitmap Encoded Safe Region (PBSR)* approach.

Bitmap Encoded Safe Region Computation: A bitmap encoded safe region represents a safe region Ψ_S for subscriber *S* using a bitmap *B* of length *n*. A bit value of 1 indicates that a predefined region (cell) belongs to the safe region; whereas a 0 bit indicates the negation. BSR techniques exhibit the following advantages: (i) for low trigger density regions, it allows for the reduction of location updates from the clients to the server when compared to other known safe region approaches (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009), (ii) it supports different granularity of safe region computations for different subscribers 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.

Consider the example in Figure 8(a). The mobile user *S* located at position P has four location triggers, each is specified by the spatial region of the trigger, denoted as $R(S, T_i)$ (i=1, 2, 3, 4). Thus the monitoring region for the mobile user *S* at point *P* is the shaded rectangle with four relevant trigger regions intersecting the grid cells. Considering a 3×3 grid overlaid on top of the monitoring region, there are only three shaded cells, which can be used as the safe region for *S* as shown in Figure 8(b). The server communicates this safe region to the client *S*. By providing each mobile client with its safe region, the server is virtually distributing the monitoring of when to wake up and check the relevant location triggers across all its mobile clients.



(a) Monitoring Region (b) Grid Bitmap Encoding (3 X 3)



(c) Grid Bitmap Encoding (9 X 9) (d) Pyramid Bitmap Encoding

Figure 8. Bitmap Encoded Safe Region Computation

Grid Bitmap Encoded Safe Region Computation Technique: The safe region for a subscriber *S* can be represented by the set of grid cells as shown in Figure 8(b). We use a grid bitmap scheme to represent the safe region of subscriber *S* as shown in Figure 8(b). The cell $C_{k,l}$ is represented by a single bit $B(C_{k,l})$. If the following condition,

$$C_{k,l} \cap \sum_{m=1}^{|A_S|} R(s, A_m) = \phi$$

is true, we set $B(C_{k,l}) = 1$, denoting that the entire cell $C_{k,l}$ belongs to the safe region Ψ_S. Otherwise we set $B(C_{k,l}) = 0$ and split $C_{k,l}$ into $U \times V$ smaller equi-sized cells. The same encoding procedure is used for each smaller cell. This bitmap encoding technique provides a compact representation for safe region Ψ_S. Figure 8(b) shows the safe region representation for the safe region of Figure 8(a) using a bitmap encoding scheme. There are no trigger regions intersecting with the three shaded cells. Thus these cells are represented by 1's in the bitmap. For those cells intersecting with some trigger regions, the corresponding bitmap cells 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 location trigger monitoring regions intersecting with it. However, this approach has two obvious drawbacks. First, the size of the grid cell constrains the grid cell-based safe region computation to be limited to the cell size and fails to take into account other finer granularity spatial areas which do not overlap with any trigger regions in the safe region computation. As visible from Figure 8(b), this safe region representation is able to represent only a small portion of the monitoring region thus providing a poor estimate of the actual safe region. Second, this approach can be expensive in terms of communication costs incurred due to more frequent broadcasting of the newly calculated safe region to each mobile client whenever a subscriber moves out of her current safe region.

Figure 8(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 in representing safe regions for two reasons: (i) it unnecessarily uses a much larger bitmap than required to represent the safe region, and (ii) different regions will have different trigger densities thus making it difficult to select a uniform grid cell size.

One approach to overcome the grid-based bitmap safe region computation is to use a *pyramid* (Samet, H., 1990) data structure, and we refer to it as the *Pyramid Bitmap Encoded Safe Region (PBSR)* approach. The pyramid approach allows for more accurate representation of the safe region and more efficient safe region computation while keeping the bitmap size small. Furthermore, the *PBSR* approach provides flexibility by allowing clients to adjust the granularity of their safe region based on their computing capability. By allowing the safe region computation for each client to be personalized according to its capability, the *BSR* computation offers greater flexibility in safe region computation by providing larger, complex safe regions for clients with higher computational capacity.

Pyramid Bitmap Encoded Safe Region Computation Technique. The pyramid representation only splits those cells in the *base* grid (level $L=0$) into $U \times V$ smaller cells when $B(C^0_{i,j}) = 0$, where U, V are system defined parameters. The process may be further repeated for several iterations to form smaller cells at each level, thus forming a pyramid data structure of height h. Figure 8(d) shows the use of a pyramid structure with $h=2$. By further splitting cells with $B(C^0_{i,j}) = 0$ into a 3×3 grid we obtain a much more accurate representation for the safe region of S. Recall that the grid-based approach, not only fails to represent the safe region accurately (3×3 grid in Figure 8(b)) when the grid cell size is large, but also fails to use a much larger bitmap (9×9 grid in Figure 8(c)) when the grid cell is small, due to the limited capacity at the mobile client and the high communication cost of broadcasting the larger bitmap to the client. In contrast, the *PBSR* approach provides flexibility in computation of the safe region while being economical in terms of bitmap size. 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 8(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 8(d). We refer interested readers to (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009) for a more detailed description of the *PBSR* approach and the algorithmic details.

4. mTrigger Client-Centric Architecture

The client-centric architecture includes the GUI interface on a mobile device, the mTrigger engine running on the mobile client, and a suite of energy-efficient location trigger evaluation techniques for optimizing both wake-up times and check times of location triggers.

In the client-centric architecture, all the necessary location trigger processing modules will reside on a mobile device such as cell phones, smart phones, and PDAs. Due to space constraints, we omit the client-centric architecture discussion and only describe briefly the optimization strategies used in our client-centric version of mTrigger. It is important to note that similar to the server side optimization, if a mobile client is far away from any of her location trigger monitoring regions, then it should be possible to compute a safe wakeup period during which the client can sleep, while guaranteeing that none of the location triggers would be missed. There are two factors that are critical in determining a wakeup period: (a) the speed of the mobile client; and (b) the size of the location trigger region.

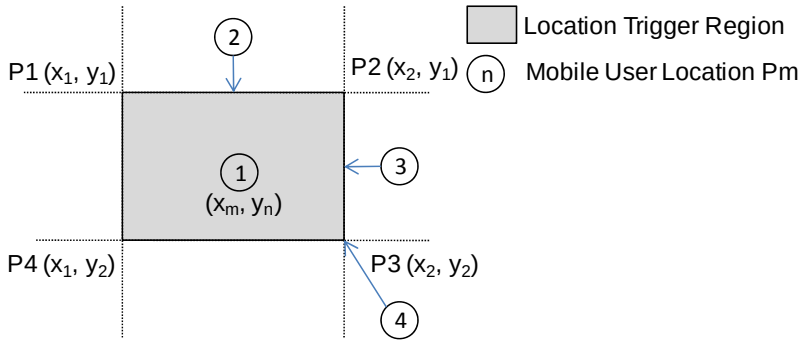


Figure 9. Possible Locations of Mobile User

To compute the speed of the mobile client and her safe period with respect to her installed location triggers, we need to measure the distance between a mobile user and the trigger region for each of her location triggers. Based on the Euclidean distance measure, there are four possible user locations P_m as described in Figure 9. We use $d_{m,R}$ to denote the distance between one of the four possible user locations (P_m) and the location trigger region *R,* and we can compute $d_{m,R}$ as follows:

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

where $D_{m,1}$, $D_{m,2}$, $D_{m,3}$, and $D_{m,4}$ represent Euclidean distance between P_m and four rectangle vertices P_1, P_2, P_3, and P_4 in R.

Based on the Euclidean distance and the expected speed, computed using maximum speed and the most recent speed, we can compute the safe period for each mobile client. Given a set of *n* location triggers installed on a client, her safe period *t* is defined by

$$t = \min(d_{m,R_1}, d_{m,R_2}, ..., d_{m,Rn}) / v_{expected},$$

where $d_{m,Rn}$ is the Euclidean distance from a mobile user's location to the location trigger region R_n and $v_{expected}$ is the expected speed.

5. Experimental Evaluation

In this section, we provide a brief evaluation of the safe period (*SP*) and safe region optimization techniques compared to periodic alarm evaluation (*PRD*) and a theoretical optimal (*OPT*) approach. The theoretical optimal approach assumes no restrictions on resource availability, namely all relevant location triggers within the monitoring region can be broadcast to the client. This implies the client is fully aware of all relevant alarms in its vicinity. We measure the performance of all approaches based on four different evaluation metrics.

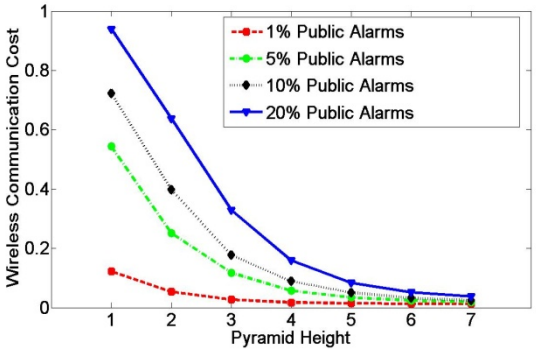
- *CPU Load/Capacity:* This factor measures the scalability of the system. It is measured as the ratio of the amount of CPU time used by the system to perform trigger processing and safe region or safe period computations to the amount of time available to the system to perform this processing. CPU load/capacity of greater than 100% indicates the failure of the system to scale to the desired configuration.
- *Wireless Communication Cost:* This is measured by the number of updates sent to the system by the mobile clients. We measure this parameter as a ratio of the communication costs required by a particular approach to the communication costs incurred by periodic trigger processing at a frequency high enough to ensure all relevant triggers are evaluated.
- *Bandwidth:* This is the downstream bandwidth (in Mbps) required by the system to communicate the safe region (or trigger information) to the clients for the safe region (or optimal) approaches.
- *Client Computation Cost:* This metric indicates the cost incurred by clients to check their position relative to the safe region in terms of average number of computations performed per client per second.

We do not measure location trigger evaluation accuracy as the parameters adopted for each processing approach ensure 100% of the location triggers are evaluated in all scenarios. The sequence of triggers to be evaluated is determined by a very high frequency trace of the motion pattern of the vehicles. We below briefly describe the experimental setup used to evaluate our system.

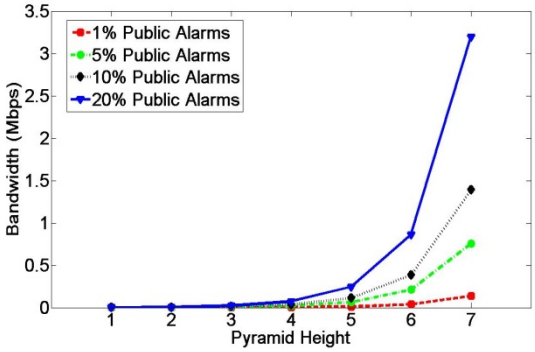
Experimental Setup: Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS, n.d.) in Spatial Data Transfer Format (SDTS (Spatial Data, n.d.)). Vehicles are randomly placed on the road network according to traffic densities determined from the traffic volume data in (Bamba, B., Liu, L., Yu, P. S., Zhang, G., & Doo, M, 2008). The simulator simulates the motion of vehicles on roads with appropriate velocity information; at intersections, vehicles may move in any direction with attached probability values. We use a map of Atlanta and surrounding region 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. Each vehicle generates a set of position parameters during the simulation which are evaluated against the generated spatial alarm information. Default values require each vehicle to generate updates with a period of less than a second for periodic processing. The default spatial alarm information consists of a set of 10,000 spatial alarms installed uniformly over the entire map region. Uniform distribution assumption is commonly considered in spatial database research. We vary the fraction of public alarms installed in the system to vary the number of alarms relevant to each client. This simulator setup allows us to the test the robustness of our framework under realistic mobility patterns.

Experimental Results: We conduct two sets of experiments detailing the performance of the *BSR* technique and present a comparison of the different approaches to location trigger evaluation.

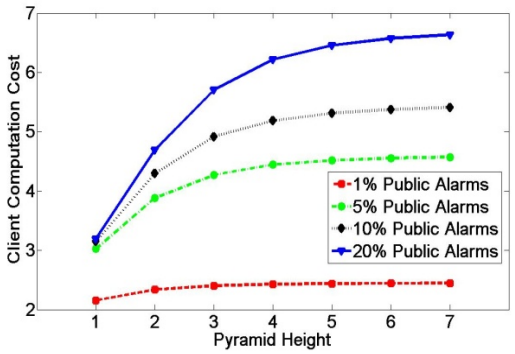
(1) Performance Evaluation of BSR Approach: This set of experiments is designed to evaluate the performance of the BSR approach. We vary the height of the pyramid from $h = 1$ (for *GBSR*) to $h = 7$ and observe the performance as shown in Figure 10. Figure 10(a) displays the wireless communication costs incurred 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 update their location 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 wireless communication costs experience a sharp drop. Another observation is that *BSR* approaches display high sensitivity to alarm density levels; the performance deteriorates sharply for higher fraction of public alarms which implies higher density of relevant alarms. On the other hand, the bandwidth required by the server to broadcast the safe regions to the clients increases with pyramid height as shown in Figure 10(b). For higher level pyramids, larger bitmaps are required to represent the safe region and hence higher bandwidth is required. For pyramid height $h = 7$, with high alarm density the downstream bandwidth requirement goes up to 3.2 Mbps, but for $h = 5$ this value remains below 250 Kbps even when the fraction of public alarms is increased to 0.2. Figure 10(c) displays the average number of computations performed per client per second to determine its position within the safe region. Clients need to perform the safe region containment detection check to determine if they need to update their position. For the *GBSR* approach the clients need to perform an average of 2-3 computations per second. This cost does not experience a significant increase with pyramid height for low fraction of public alarms. For higher fraction of public alarms the costs rise to 6-7 computations per second for a pyramid of height $h = 7$. As seen from Figure 10(d), for low pyramid height, safe region computation costs are low as relatively simpler computations are involved. On the other hand, alarm processing costs are high as a large number of updates are received from clients. On increasing pyramid height, alarm processing costs drop due to fewer client position updates. The safe region computation costs increase due to high complexity of safe region computation. Even despite the fewer number of safe region computations being performed at a higher pyramid height, the increase in cost of a single safe region computation is such that a



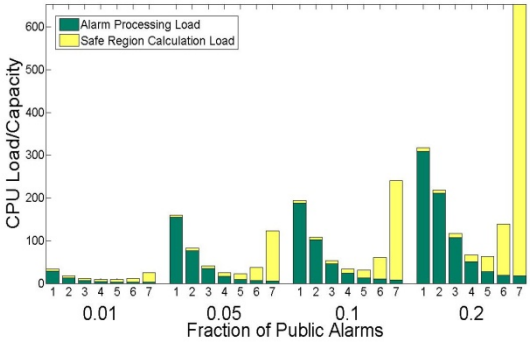
(a) Wireless Communication Cost



(b) Bandwidth (Mbps)



(c) Client Computation Cost

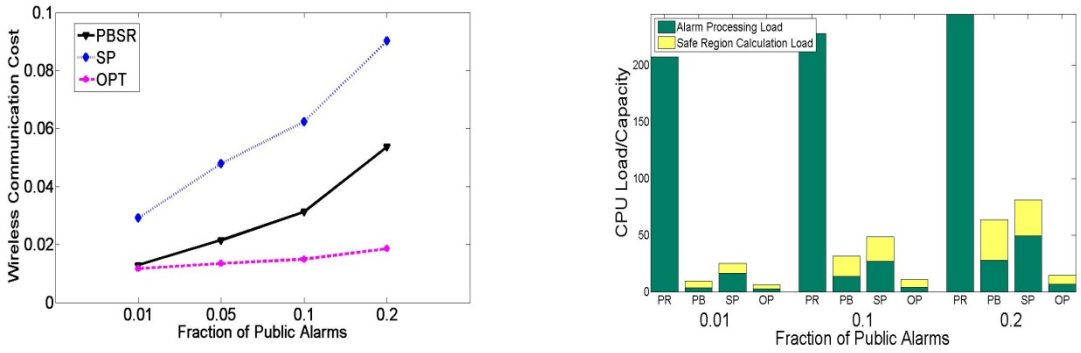


(d) CPU Load to Capacity Ratio

Figure 10. Experimental Results for *BSR* Approach

net increase in safe region computation load is experienced. However, this cost can be significantly offset by using precomputed bitmaps for public alarms as described earlier. For $h = 4$ or $h = 5$, the overall CPU load is at its lowest point. With increasing the fraction of public alarms the system experiences an increase in overall CPU load.

(2) Performance Comparison of Alarm Evaluation Techniques: Now we compare the performance of the safe region approach (*PBSR)* with periodic processing, safe period-based processing and the optimal approach. As seen in Figure 11(a), the safe region approach (*PBSR with h=5*) incurs very low wireless communication costs. Periodic processing requires clients to transmit each location update to the server incurring a wireless cost of 1 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 approach. This is largely due to the pessimistic assumptions required to ensure that the safe period approach evaluates all triggers with a 100% success rate. For lower trigger density levels the gap between the optimal and safe region approach is much lower. The optimal approach would require clients to transmit updates only when the spatial constraints for one or more relevant location triggers are met. The CPU load experienced by each approach is as shown in Figure 11(b). The periodic approach (PR) has much higher trigger processing costs as each update needs to be processed by the client and the CPU load does not scale. The processing load does not increase much at higher trigger densities, as each update is processed by this approach for all fractions of public triggers. The safe region approach, denoted by *PB* in the figure, experiences lower CPU load due to a much lower trigger processing load. With an increasing fraction of public alarms, the safe region computation, as well as the trigger processing load, rises. However, the total load incurred by the system is much lower than the periodic approach for all configurations. The safe period (SP) approach experiences a higher CPU load compared to the safe region approaches. This is a direct result of the larger number of location updates that need to be processed by the safe period approach. Results for the optimal approach are plotted to show that the safe region approaches do not incur much higher CPU load except for the situation when the fraction of public alarms is very high.



(a) Wireless Communication Cost

(b) CPU Load to Capacity Ratio

Figure 11. Experimental Comparison of Different Alarm Evaluation Techniques

6. Related Work

There are a number of research areas related to the mTrigger system, including Human Computer Interaction (HCI), Spatial-temporal databases, Geographical Information Systems (GIS), games programming, to name a few.

In the HCI area, location reminder systems have been studied by several projects (Dey, A., & Abowd. G., 2000), (Kim, S., Kim, M., Park, S., Jin, Y., & Choi, W., 2004), (Ludford, P., Frankowski, D., Reily, K., Wilms, K., & Terveen, L., 2006). , (Marmasse, N., & Schmandt, C., 2000), (Sohn, T., Li, K., Lee, G., Smith, I., Scott, J., & Griswold, W., 2005), focusing on usability of such applications. However, all existing work in the HCI area has been conducted from the usability perspective of the location reminder systems and applications. Geominder (Geominder, n.d.). and Naggie (Naggie 2.0, n.d.) are location reminder systems providing useful location reminder services using cell tower ID and GPS technology, respectively. With the development of mobile platforms like Android (Android, n.d.) and iPhone SDK (Apple Developer Connection, n.d.), we see similar applications made available on a multitude of smart phones. However, none of these existing approaches deal with the system level performance optimization issues, which we believe are critical for wide deployment of any location trigger-based systems and applications.

In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand. User defined triggers can be initiated when new relevant information which is of personal interest to the user is detected by the system (Bazinette, V., Cohen, N., Ebling, M., Hunt, G., Lei, H., Purakayastha, A., Stewart, G., Wong, L., & Yeh, D., 2001), 23. In addition to monitoring continuously changing user information needs, the mTrigger system also needs 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 the continuous monitoring of moving objects (Jensen, C. Lin, D., & Ooi. B., 2004), (Mokbel, M., Xiong, X., & Aref, W., 2004), (Yu, X.,

Pu, K., & Koudas, N., 2005). Some work exists on monitoring continuous queries, which applies the concept of safe region directly or indirectly (Cai, Y., & Hua, K., 2002), (Hu, H., Xu, J., & Lee, D., 2005). Spatial alarms differ from this work, as they do not demand periodic evaluation or reevaluation like continuous queries; instead they require one shot evaluation which should result in a trigger when the alarm conditions are satisfied. Once a trigger is activated, no further evaluation of the trigger is required. Our work is focused on determining the opportune moment for evaluating spatial alarms relevant to a client by seeking cooperation at the client end.

Furthermore, numerous works have dealt with the problem of energy conservation in mobile devices (Flautner K., & Mudge. T., 2002), 18, (Mudge, T., 2001). To the best of our knowledge, none have systematically addressed the processing of location triggers using an event-based framework. mTrigger is the first system that utilizes the concept of location triggers as an abstraction to model a variety of location-based monitoring and notification services, such as spatial alarms, (Bamba, B., Liu, L., Yu, P. S., & Iyengar, A., 2009), (Murugappan, A., & Liu, L., 2008), (mGraffiti, n.d.), and location-based advertisements (GPS Daily, 2008).

Other areas which deal with spatial regions are Geographic Information Systems (GIS) and game programming. In principle, Geographical Information Systems (GIS) do not deal with mobile objects and their movement characteristics. Instead, GIS focuses on map visualization and spatial data analysis.

In the area of game programming, detailed animated objects interact not only with their complex environments but also with each other. Real-time collision detection Ericson, C. (2005) is required in order to maintain a believable simulation status. It is essential to consider the approximate shapes of complex objects for collision detection in such environments. Additionally, rendering of objects is essential to maintain reasonable frame rates. Dead reckoning optimizations are commonly applicable to handle network delay impact (Baughmann, N. E., & Levinne, B. N., 2001) in such environments. Our event-based system does not demand such continuous knowledge of position information of mobile clients which makes the presented optimizations feasible. Thus, techniques for scaling location trigger processing, such as safe period and safe region are critical optimizations for location dependent event monitoring systems and applications.

7. Conclusion and Future Work

We have presented mTrigger – an event-based framework for providing a scalable infrastructure for supporting location-based event systems and applications in the future pervasive computing environment. The mTrigger development has two unique features. First, it provides an event-based location trigger framework for supporting large scale location trigger based applications through mTrigger plug-in modules. We demonstrate the usage and the benefits of mTrigger middleware framework through a number of mobile applications developed for the Georgia Tech campus. Second, we develop the safe period and safe region based optimization techniques as one of the building blocks of mTrigger's design for providing scalable and highly efficient location trigger evaluation. Our experimental

evaluation shows that the mTrigger framework and its safe period and safe region optimizations can significantly reduce server processing load and application development cycle.

Our ongoing and future work continues along three directions. First, we are actively studying the optimization techniques for location triggers of mobile users traveling on road networks, especially how road network and travel time constraints can be utilized to optimize the processing of location triggers and enhance accuracy and performance of event based applications built on top of location triggers, such as location based advertisements, location based entertainment, and location-based tracking. Second, we are currently developing concrete mobility-based user group formation methods to be used as the building blocks for co-operative location trigger evaluation in decentralized geographical overlay networks (Zhang, J., Zhang, G., & Liu, L., 2007). Third but not the least, we are interested in location privacy and location security issues (Bamba, B., Liu, L., Pesti, P., & Wang, T., 2008), (Gedik, B., & Liu, L., 2008), (Wang, T., & Liu, L., 2009) for safeguarding user privacy in the location trigger supported event systems and applications.

Acknowledgement:

This work is partially sponsored by grants from NSF CISE NetSE program, CyberTrust program, an IBM faculty award, an IBM SUR grant, and an Intel research council grant. The authors are also grateful to many discussions with members of the Distributed Data Intensive Systems Laboratory (DiSL) at Georgia Institute of Technology.

References

Android. (n.d.). *An Open Handset Alliance Project*. Retrieved from http://code.google.com/android/

Apple Developer Connection (n.d.). *iPhone SDK.* Retrieved from http://developer.apple.com/iphone/program/download.html

Geominder (n.d.). *Geominder.* Retrieved from http://ludimate.com/products/geominder/

Lost in Klaus (n.d.). *Lost in Kluas.* Retrieved from http://www.cc.gatech.edu/~ulee3/LostInKlaus/

mGraffiti (n.d.). *mGraffiti.* Retrieved from http://www.cc.gatech.edu/projects/disl/mGraffiti/

Naggie 2.0 (n.d.). *Naggie 2.0: Revolutionize Reminders with Location!* Retrieved from http://www.naggie.com/

GPS Daily (2008, February 6). *NXP Fuels Rise of Mobile Location-Based Services*. Retrieved from http://www.gpsdaily.com/reports/NXP_Fuels_Rise_Of_Mobile_Location_Based_Services_999.html

Spatial Data (n.d.). *Spatial Data Transfer Format*. Retrieved from http://www.mcmcweb.er.usgs.gov/sdts/

USGS (n.d.). *U.S. Geological Survey*. Retrieved from http://www.usgs.gov

Bamba, B., Liu, L., Yu, P. S., Zhang, G., & Doo, M. (2008, December). Scalable Processing of Spatial Alarms. In *Proceedings of 15th International Conference on High Performance Computing*. Bangalore, India.

Bamba, B., Liu, L., Yu, P. S., & Iyengar, A. (2009, June). Distributed Processing of Spatial Alarms: A Safe Region-based Approach. In *Proceedings of IEEE Int. Conf. on Distributed Computing*, June 22-26, Montreal, Quebec, Canada.

Bamba, B., Liu, L., Pesti, P., & Wang, T. (2008, April). Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid. In *Proceedings of 17th International World Wide Web Conference* (WWW). Beijing, China

Baughmann, N. E., & Levinne, B. N. (2001). Cheat-proof playout for centralized and distributed online games, In *Proceedings of IEEE INFOCOMM*.

Bazinette, V., Cohen, N., Ebling, M., Hunt, G., Lei, H., Purakayastha, A., Stewart, G., Wong, L., & Yeh, D. (2001). *An Intelligent Notification System*. IBM Research Report RC 22089 (99042).

Cai, Y., & Hua, K. (2002). An Adaptive Query Management Technique for Efficient Real-Time Monitoring of Spatial Regions in Mobile Database Systems. In *Proceedings of the IEEE IPCCC*, (pages 259–266).

Dey, A., & Abowd. G. (2000). CybreMinder: A Context-Aware System for Supporting Reminders. In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing*, (pages 172–186).

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

Flinn, J., & Satyanarayanan, M. (1999). Energy-Aware Adaptation for Mobile Applications. In Proceedings of *SOSP*, pages 48–63, 1999.

Gedik, B., & Liu, L. (2008, January). Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. In *Proceedings of IEEE Transactions on Mobile Computing* (Vol. 7, No. 1, pp. 1-18).

Hu, H., Xu, J., & Lee, D. (2005). A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In *Proceedings of ACM SIGMOD*, 2005.

Jensen, C. Lin, D., & Ooi. B. (2004). Query and Update Efficient B+-Tree based Indexing of Moving Objects. In *Proceedings of VLDB*, (pages 768–779).

Kim, S., Kim, M., Park, S., Jin, Y., & Choi, W. (2004). Gate Reminder: A Design Case of a Smart Reminder. In *Proceedings of the Conference on Designing Interactive Systems*, (pages 81–90).

Liu, L. Pu, C., & Tang, W. (2000). WebCQ - Detecting and Delivering Information Changes on the Web. In *Proceedings of CIKM*, (pages 512–519).

Ludford, P., Frankowski, D., Reily, K., Wilms, K., & Terveen, L. (2006). Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications. In *SIGCHI Conference on Human Factors in Computing Systems*, (pages 889–898)

Marmasse, N., & Schmandt, C. (2000). Location-Aware Information Delivery with Com-Motion. In *Proceedings of HUC*, (pages 157–171).

Mokbel, M., Xiong, X., & Aref, W. (2004). SINA: Scalable Incremental Processing of Continuous Queries in Spatio-Temporal Databases. In *ACM SIGMOD*, (pages 623–634).

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

Murugappan, A., & Liu, L. (2008, June 30-July 2). An Energy Efficient Approach to Processing Spatial Alarms on Mobile Clients. In *Proceedings of the ISCA 17th International Conference on Software Engineering and Data Engineering* (SEDE-2008).

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Reading, MA: Addison-Wesley.

Sohn, T., Li, K., Lee, G., Smith, I., Scott, J., & Griswold, W. (2005). Place-Its: A Study of Location-Based Reminders on Mobile Phones. In *Proceedings of UbiComp*.

Yu, X., Pu, K., & Koudas, N. (2005). Monitoring k-Nearest Neighbor Queries over Moving Objects. In *Proceedings of ICDE*, (pages 631–642)

Spatial Alarms Project (n.d.). *Spatial Alarms Project*. Retrieved from http://www.cc.gatech.edu/projects/disl/SpatialAlarms/

mTrigger (n.d.). *mTrigger Project*. Retrieved from http://www.cc.gatech.edu/projects.disl/mTriggers/

Ericson, C. (2005). *Real-Time Collision Detection*. San Francisco: Morgan Kaufmann.

Wang, T., & Liu, L. (2009, September). Privacy-Aware Mobile Services over Road Networks. In *Proceedings of the 35th International Conference on Very Large Data Bases*. PVLDB 2(1): 1042-1053 (2009).

Zhang, J., Zhang, G., & Liu, L. (2007, September). GeoGrid: A Scalable Location Service Network. In *Proceedings of the 27th IEEE International Conference on Distributed Computing Systems*.