Energy-Efficient Processing of Spatial Alarms on Mobile Clients

Anand Murugappan
College of Computing
Georgia Institute of Technology
Atlanta, USA
anandm@cc.gatech.edu

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

Abstract

In this paper we present an energy efficient framework for processing spatial alarms on mobile clients, while maintaining low computation and storage costs. Our approach to spatial alarms provides two systematic methods for minimizing energy consumption on mobile clients. First, we introduce the concept of safe distance to reduce the number of unnecessary mobile client wakeups for spatial alarm evaluation. This mechanism not only reduces the amount of unnecessary processing of the spatial alarms but also significantly minimizes the energy consumption on mobile clients, compared to periodic wakeups, while preserving the accuracy and timeliness of the spatial alarms. Second, we develop a suite of techniques for minimizing the number of location triggers to be checked for spatial alarm evaluation upon each wakeup. This further reduces the computation cost and energy expenditure on mobile clients. We evaluate the scalability and energy-efficiency of our approach using a road network simulator. Our client based framework for spatial alarms offers significant improvements on both system performance and battery lifetime of mobile clients, while maintaining high quality of spatial alarm services, especially compared to the conventional approach of periodic wakeup and checking all alarms upon wakeup.

1 Introduction

Many on a daily basis use time based alarms. Spatial alarms extend the very same idea to location-based triggers, which are fired whenever a mobile user enters the spatial region of the location alarms. Spatial alarms provide critical capabilities for many mobile location based applications ranging from personal assistants, inventory tracking to industrial safety warning systems. In this paper we present our architecture for energy efficient processing of spatial alarms on mobile clients, while maintaining low computation and storage costs. We present two systematic methods that can progressively minimize the amount of energy consumption on mobile clients for all types of spatial alarms. The first method utilizes the concept of safe distance to reduce the number of unnecessary wakeups on mobile clients for spatial alarm evaluation. By enabling mobile clients to sleep for longer intervals of time in the presence of active spatial alarms, we show that our safe distance techniques can significantly minimize the energy consumption on mobile clients compared to periodic wakeups, while preserving the accuracy and timeliness of spatial alarms. The second mechanism focuses on alarm checks upon each wakeup. We develop a suite of techniques for minimizing the number of location triggers to be checked upon each wakeup for different types of spatial alarms. This allows us to further reduce the computation cost and energy expenditure on mobile clients. Our experimental evaluation using a road network simulator shows that our spatial alarms middleware architecture offers significant improvements on battery lifetime of mobile clients, while maintaining high quality of spatial alarm services compared to the conventional approach of periodic wakeup and checking all alarms upon a wakeup. In addition to energy efficiency, another important goal of spatial alarm processing that we consider is to maintain the low or zero alarm misses.

2 System Model

A spatial alarm consists of three components: the spatial region on a two-dimensional geographical plane, the action to be taken upon firing of the alarm, and the alarm termination condition, usually a temporal event such as time point or time interval. The spatial regions used in spatial alarms can be of any shape. We capture each of such spatial 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 top-left and bottom-right vertices of the bounding rectangle. Without loss of generality, in the rest of the paper, we simply assume that each mobile client can install n spatial alarms

($n \geq 0$) and all spatial alarms are expressed by a rectangle spatial region, denoted by A_i for $1 \leq i \leq n$. In our first spatial alarm client middleware prototype, we use a system supplied default spatial range in the absence of spatial region specification of a user-defined alarm. Each mobile client can install as many spatial alarms as the user wishes over the geographical area of interest. Multiple mobile clients can set spatial alarms on the same locations.

3 System Overview

Spatial alarms differ from spatial location queries in a number of ways. First, spatial queries such as "tell me the gas stations within 10 miles on the highway 85 north" require continuous evaluation of the queries as the mobile client moves on the highway 85 north. However, a spatial alarm such as "notify me whenever I am 5 miles away from this particular dry cleaning store (marked on the map)", only requires the alarm to be evaluated when the mobile client moves to a region that is within 5 miles of the specific dry cleaning store. Thus, it is of no use to wake up a mobile client if she is 30 miles away from the dry cleaning store. Clearly, the movement patterns of the mobile client and the distance from the current location of a mobile client to all her alarms are the two critical factors that affect when the mobile client needs to wakeup and what alarms need to be checked upon each wakeup. Thus one can optimize the spatial alarm processing by devising more energy efficient algorithms.

Mobile devices conserve energy by spending most of their time in a low energy state such as sleep mode. Hence one of the critical design objectives for client middleware architecture is to minimize the number of device wakeups in spatial alarm processing. For instance, the 206 MHz Itsy [2] pocket computer spends 540mW power in the *System Idle, 0% processor idle* state, spends 100mW power in the *System Idle, 95% processor idle* state while in the *Sleep mode* it just spends 8.39mW power (which is about 64 times lesser than the 0% processor idle case). It is interesting to note that mobile devices like in the case of Itsy computer [2] have a battery lifetime of only 3.8 hrs when running in the high energy *System Idle, 95% processor idle* state, while in the *Sleep mode* the lifetime is as high as 279 hrs.

The conventional approach for implementing a location based reminder system is to wake up the device and check the alarm conditions periodically. If the period is too large the mobile device might miss alarms since there may be situations where the mobile client passes through the 'alarm area' while asleep (between periodic checks). Hence, to reduce the number of alarm misses, the wakeup period would have to be kept small enough. The smallest wakeup period can be set using the location update frequency (e.g., GPS sampling period). Clearly, the periodic check approach would be very energy inefficient. Also it is important to note that if the mobile client is far away from any of her alarms then depending on the maximum speed of the client, it is possible to sleep for longer durations of time and still guarantee that none of the alarms would be missed.

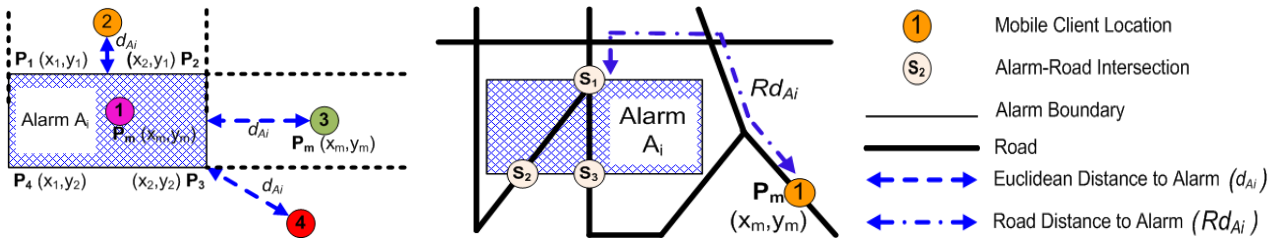
4 Minimizing Device Wakeups

Our architecture for spatial alarm processing consists of two phase optimizations. In this section we discuss the phase one optimization strategies that minimize the number of device wakeups. Apart from the high energy consumption, an important problem with the periodic wakeup approach is that it is hard to estimate how frequently the device should wakeup to ensure no alarms will be missed. Two factors that are critical in determining such a frequency: (a) The speed of the mobile client; and (b) the size of the spatial alarm region. Unless the frequency is set to be extremely high (close to the location update frequency), it would always be possible to introduce cases where alarms can be missed by having alarms of the size smaller than the distance traveled by the mobile client between two consecutive wakeups. Thus the key challenge is to determine the right time for mobile clients to wakeup in terms of energy efficiency and alarm accuracy and given a location update, how to determine the subset of alarms that should be checked to conserve energy while maintaining zero alarm misses.

With both the problem of guaranteed alarm delivery and that of energy conservation in mind, we propose four optimization strategies to estimate the safe period and use aperiodic wakeup of the mobile client based on (a) the distance of the client to all her alarms and (b) the travel speed of the mobile client.

4.1 Measuring the Distance to Alarm

There are two most commonly used methods for measuring the distance from a mobile client's current location to an alarm. They are (a) Euclidean Distance and (b) Road Network Distance. The *Euclidean Distance* approach is simpler and requires much lesser data but may at times underestimate the time to sleep before the next wakeup. The *Road Network Distance* measure offers a more accurate estimate of the distance from a mobile client's current location to the spatial region of the alarm, but it introduces additional overhead with handling the road network map data. We



(a) Euclidean Alarm Distance (b) Road Network Alarm Distance (c) Legend

Figure 1: Distance to Alarm from Mobile Client

propose techniques to mitigate this additional overhead by dividing the original map into tiles and selectively downloading relevant tiles to a mobile client.

Given a spatial alarm A_i with rectangular spatial alarm region represented by four vertices of the rectangle: (P_1, P_2, P_3, P_4) where $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_1)$, $P_3 = (x_2, y_2)$ and $P_4 = (x_1, y_2)$. Let the mobile client be at P_m represented by the coordinates (x_m, y_m), then the Euclidean distance from P_m to the alarm region of A_i, denoted by d_{A_i}, can be computed by considering four cases. *Case 1:* when the mobile device is within the alarm boundaries the distance to the alarm is zero; *Case 2:* when the mobile device is within the y scope (represented using dotted lines in Figure 1 a)) the distance is the shortest of the distances to alarm edges parallel to the x axis from the mobile client; *Case 3:* when the mobile device is within the x scope the distance is the shortest of the distances to alarm edges parallel to the y axis from the mobile client; and *Case 4:* when the mobile device is outside both the x and y scopes, then the distance is the minimum of the Euclidean distances to the four vertices. The four cases can be formally defined as follows:

$$d_{A_i} = \begin{cases} 0 & x_1 \le x_m \le x_2 \\ & and \;\; y_1 \le y_m \le y_2 \\ min(|x_m - x_1|, |x_m - x_2|) & y_1 \le y_m \le y_2 \; only \\ min(|y_m - y_1|, |y_m - y_2|) & x_1 \le x_m \le x_2 \; only \\ min(D_{m1}, D_{m2}, D_{m3}, D_{m4}) & otherwise \end{cases}$$

Where $D_{m1}, D_{m2}, D_{m3}, D_{m4}$ denote the Euclidean distance from P_m to the four rectangle vertices P_1, P_2, P_3, P_4 respectively. The distance function $D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is used to compute the Euclidean distance between two points P_i and P_j.

One of the main weaknesses of the *Euclidean Distance* measure is that the estimated distance is often shorter than the actual distance that the mobile client would have to travel to get to the spatial region of interest of a given alarm due to the underlying traversal restrictions imposed by the road network. The *Road Network Distance* measure uses the Dijkstra's shortest path algorithm [3] to estimate the distance from the mobile client's current location to an alarm as shown in Figure 1 (b). The underlying road network is represented by the solid line and the mobile client is represented by a shaded circle labeled by 1. Since the mobile client is restricted to move along the roads, the only places where it can enter the alarm area would be the points of intersection of the alarm with the roads, denoted by S_1, S_2, S_3 in Figure 1 (b).

4.2 The Aperiodic Wake-Up Algorithms

In this section, we introduce the concept of safe period based on the distance function and the speed function, and present four safe period based wakeup algorithms by combining the distance functions with the speed functions.

$$T_{sleep} = \min(d_{A_1} ... d_{A_i} ... d_{A_n})/v_{max} \quad (1)$$

$$T_{sleep} = \min(d_{A_1} ... d_{A_i} ... d_{A_n})/v_{expected} \quad (2)$$

$$T_{sleep} = \min(Rd_{A_1} ... Rd_{A_i} ... Rd_{A_n})/v_{max} \quad (3)$$

$$T_{sleep} = \min(Rd_{A_1} ... Rd_{A_i} ... Rd_{A_n})/v_{expected} \quad (4)$$

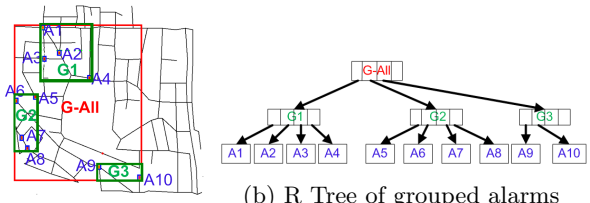
where T_{sleep} is time duration for which the mobile client can sleep without potentially missing delivery of any alarm, n is the total number of alarms installed on the mobile client, d_{A_i}, Rd_{A_i} are the euclidean and road network distances from the mobile client's current location to the i^{th} spatial alarm A_i $(1 \le i \le n)$ and v_{max}, $v_{expected}$ are the maximum and expected travel speeds of the mobile client as defined below.

$$v^p_{expected} = 0 \quad (5)$$

$$v^c_{expected} = \beta * \frac{D(l_c, l_p)}{t_c - t_p} + (1 - \beta) * v^p_{expected} \quad (6)$$

$$v_{expected} = \alpha * v^c_{expected} + (1 - \alpha) * v_{max} \quad (7)$$

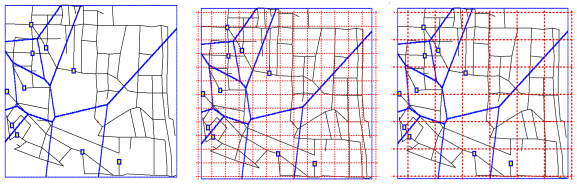
where $v^p_{expected}$, $v^c_{expected}$, $v_{expected}$ are the previous, the current, and the future *expected* travel speed of the mobile client respectively, t_c and t_p represent the current and previous time instances, l_c and l_p represent the current and the previous location of the mobile client at time instances t_c and t_p respectively and D is the distance function.



(a) Alarm Grouping

(b) R Tree of grouped alarms

Figure 2: Alarm Grouping by Spatial Proximity



(a) Voronoi Regions

(b) Small grid cells

(c) Large grid cells

Figure 3: Alarm Grouping by using grid cells

Safe Distance with Max Speed in Equation 1 defines the safe distance of a mobile client to each of her spatial alarms by combining the *Euclidean Distance* function and the maximum. At each wakeup, the mobile client will perform two tasks - (a) process the spatial alarms installed on the client (b) estimate the safe period (i.e. the time to sleep) denoted by T_{sleep}, before the next wakeup. **Safe Distance with Expected Speed** in Equation 2 exploits the fact that not all mobile clients might travel at the maximum possible speed at all times. Since 'expected speed' would ideally be lesser than the maximum speed, the number of device wakeups can be further reduced at the expense of potentially missing some alarms. **Safe Road Distance with Max Speed** in Equation 3 exploits the limitations imposed on the movement of the mobile client by the underlying road network while ensuring no alarms are missed. **Safe Road Distance with Expected Speed** in Equation 4 is a natural extension and combines the advantages of the 'expected speed' approach with the 'road distance' approach.

5 Minimizing Alarms Checked

In this section we argue that the approach of *check all alarms upon each wakeup* is naive and wasteful of resources. We describe two approaches to minimize the number of alarm checks per wakeup and show how these approaches can reduce the computation cost and the energy consumption involved in alarm checks. In the first approach, we group spatial alarms that are in close spatial proximity, in a hierarchical fashion. Alarm Checking happens in groups, thus minimizing the overall number of alarm checks performed per wakeup. In the second approach, we divide the geographical area of interest into Voronoi regions based on Euclidean distance to the alarms with each region storing information that can quickly identify the nearest alarm in the vicinity. Upon each wakeup, alarm checks are performed only against the 'nearest' alarm by looking up the information in the Voronoi region in which the mobile client currently resides.

5.1 Hierarchical Grouping of Alarms

When the geographical area in which a mobile client installs her alarms is big, the number of alarms installed is large and distributed across the entire area of interest, checking all alarms upon each wakeup is not only unnecessary but also a clear waste of resources. We first propose to group spatial alarms based on their spatial proximity and check the alarms in selected groups upon each wakeup. The grouping process proceeds in two steps. First, all alarms need to be divided into groups based on spatial proximity with each group associated with a spatial region. Only when the mobile client moves into the region marked by a group, the spatial alarms/subgroups within that group will be checked, and all other alarms/subgroups belonging to the other groups are eliminated from alarm checking, leading to significant saving in terms of computational cost and energy. We use the R Tree [7] algorithm to perform the alarm grouping in a hierarchical fashion as shown in Figure 2.

The R-Tree based alarm grouping algorithm is effective in terms of energy saving and resource usage in general and especially it can handle well, the situations where the mobile client continuously adds new spatial alarms into the client middleware system as she moves on the road. However, if the number and the location of the spatial alarms remain unchanged for long duration of time, we can utilize the Voronoi diagram [1] to devise a more efficient alarm group algorithm. We below present two such algorithms, one uses Voronoi regions, called nearest alarm check algorithm, and the other uses the Network Voronoi diagrams, called the road network nearest alarm check algorithm.

5.2 Checking Nearest Alarm Only

The *checking nearest alarm only* algorithm is suitable for the scenarios where the number and location of alarms remain unchanged for long duration of time and no addition or removal of alarms are issued by the mobile client. The *Nearest Alarm Only* optimization consists of two phases. In the first phase, the two dimensional geographical area of interest is divided into

grid cells of equal size. Then the Voronoi diagram is overlaid on top of the grid with Voronoi Regions [1, 5] such that each Voronoi Region has a single nearest alarm, as shown in Figure 3(a). To facilitate the search for the nearest alarms for a given mobile client location, we build a grid cell based dense index, in which each cell contained in a Voronoi region will point to the spatial alarm of that region, and each cell that overlaps with k Voronoi regions ($1 < k < n$) will contain k spatial alarms, each corresponding to one of the k Voronoi regions.

In the second phase, upon wakeup the mobile client uses her current location to locate the grid cell in which she resides and it takes only O(1) to lookup the nearest alarm in the case where the grid cell of the client is contained in a Voronoi region. In the situation where the mobile client is at boundaries of k Voronoi regions ($1 < k < n$), the grid cell in which the client resides will point to k spatial alarms, all are qualified to be the 'nearest' alarms. In this scenario all the Voronoi regions overlapped with the current location of the mobile client need to be considered, and the alarm check will be performed against the 'nearest' alarm in each of these overlapped Voronoi regions. Clearly, this approach greatly reduces the time to lookup the relevant alarms to be checked, although it is only applicable in the specific scenarios where alarms are not frequently removed or added (since computing the Voronoi diagram for the entire geographical area of interest each time a new alarm is added or an existing alarm is removed can be quite expensive).

An alternative way to improve the storage cost is to use the Network Voronoi Diagrams [4, 6]. The checking nearest alarm with road network algorithm consists of two phases. In the first phase, we need to build Network Voronoi diagram that can partition the road network among the alarm nodes. In the second phase, each of the nodes on the road network graph is associated with a list of 'nearest alarms' based on the road network distance.

6 Experimental Evaluation

6.1 Estimating Energy and Battery Life

In order to measure the energy consumed and battery lifetime for various wakeup algorithms and alarm check algorithms, we use the device energy values corresponding to the itsy pocket computer [2] given in the energy parameter table of Figure 6 as our reference model. One can express the total energy E_t consumed as a function of the number of wakeups (N_w), the minimum time duration per wakeup (T_m), the total time duration (T_t), the power consumption of the mobile device while awake (P_{100}), idle (P_i) and asleep (P_s), and the *Alarm Check Ratio* (C_r) which represents the ratio of the number of total actual alarm checks performed to the maximum total checks that can be performed $N_{A_{MAX}}$ during the minimum wakeup durations T_m as follows:

$$E_t = N_w T_m (C_r P_{100} + (1 - C_r) P_i) + (T_t - N_w T_m) P_s \quad (8)$$

If C_b is the battery capacity, T_t is the total time of the experiment, and E_t represents the energy spent during T_t then the battery lifetime of a mobile client, denoted by T_b, can be calculated by the following equation:

$$T_b = \frac{C_b \times T_t}{E_t} \quad (9)$$

Due to lack of space we omit the derivation. For details please refer to our technical report [9].

6.2 Experimental Results

In this section we present a set of experimental results. Our results demonstrate three important conclusions. First, the proposed wakeup and alarm check algorithms offer significant (up to 6.4 times) reduction in terms of energy consumption in comparison to the naive approach with periodic wakeups followed by checking all alarms per wakeup. Second, the *Safe Road Distance with Max Speed* wakeup algorithm offers the maximum energy conservation with 100% alarm delivery guarantee. Third but not the least, the *alarm grouping* check algorithm is the most flexible among the alternative alarm check strategies and offers significant reduction in terms of energy consumption, and significant improvement (up to 50%) in battery life when the number of alarms is high. The *Nearest-only Alarm* and the *Network Nearest-only Alarm* algorithms offer even better improvements in terms of energy consumption but have limited applicability and can only be used in those cases where addition and removal of alarms are less frequent and low in numbers.

6.2.1 Effect on Overall Energy Consumption

Figure 5(left and center-left) compare the variation of Wake-Up frequencies with the increase in number of alarms. As one would expect the Periodic Wake-Up strategy is indifferent to the increase in the number of alarms. When the mobile client has only one alarm per map tile, the proposed wakeup algorithms reduce the number of wakeups by almost an order of magnitude. The two *Safe Road Distance* algorithms perform better than the *Safe Distance* algorithms. The wakeup algorithms that use *expected speed* reduce the frequency of wakeups further.

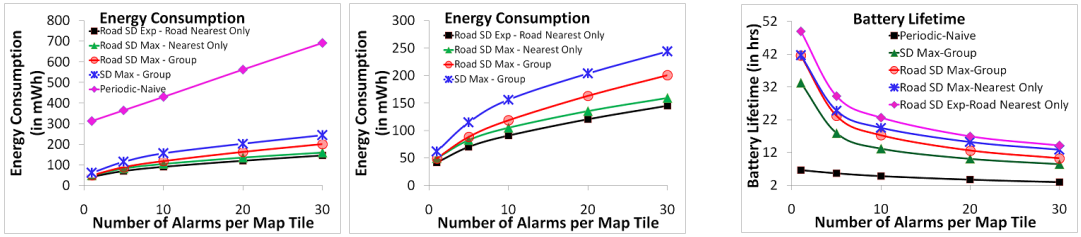


Figure 4: Performance of WakeUp-Check strategy combinations

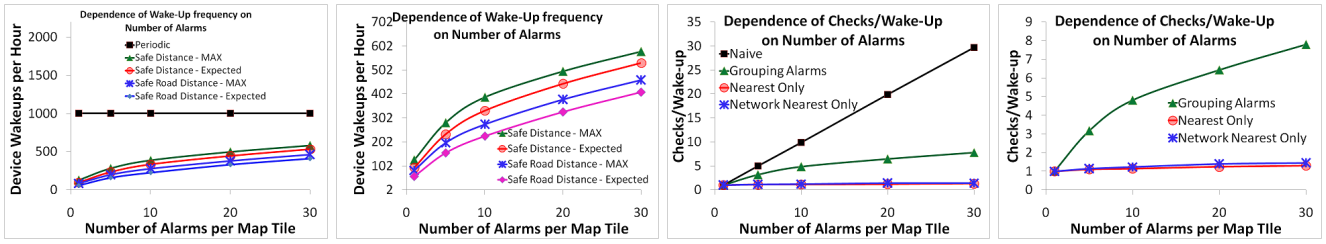


Figure 5: Effect of Changes in Number of Alarms

7 Related Work and Conclusion

[10] introduces the idea of *Query Indexing* and *Velocity Constrainted Indexing* solves the more generic problem of continous queries. However, given the generic nature of the problem it does not exploit aspects related to the road network and optimizations such as query grouping. Certain others like [8] focused on identifying the user's *logical* location by mapping information such as IP Address, latitude and longitude information in GPS readings. The logical location could then be used to automatically connect the mobile client to local resources such as printers.

We have presented an energy efficient framework for processing spatial alarms on mobile clients. Our experiments show that the proposed client-based spatial alarms architecture offers significant improvements in battery lifetime of mobile clients, while maintaining high quality of spatial alarm services compared to the conventional approach of periodic wakeup and checking all alarms upon each wakeup.

Parameter	*Default*
Number Of Alarms (N_A)	10
Clustering Factor	2
Simulation Duration (T_t)	3 *hrs*
100% *CPU Power* (P_{100})	540 *mW*
IdlePower (P_i)	100 *mW*
SleepPower (P_s)	8.39 *mW*
Minimum Wake Duration (T_m)	0.001 *hrs*
Battery Capacity (C_b)	2052*mWh*

Figure 6: Simulator Parameters

Acknowledgements

This work is partially supported by grants from NSF IIS SGER, NSF CSR, and NSF CyberTrust, an IBM SUR grant, and an IBM faculty award.

References

[1] F. Aurenhammer. Voronoi diagrams a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[2] J. F. Bartlett, L. S. Brakmo, K. I. Farkas, W. R. Hamburgen, T. Mann, M. A. Viredaz, C. A. Waldspurger, and D. A. Wallach. The itsy pocket computer, 2000.

[3] E. W. Dijkstra. A note on two problems in connection with graphs., 1959.

[4] M. Erwig. The graph voronoi diagram with applications. In *Networks*, pages 156–163, 2000.

[5] S. Fortune. Voronoi diagrams and delaunay triangulations. pages 377–388, 1997.

[6] M. Graf and S. Winter. Netzwerk-voronoi-diagramme. sterreichische zeitschrift fr vermessung und geoinformation. pages 166–174, 2003.

[7] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM Press.

[8] J. Heidemann and D. Shah. Location-aware scheduling with minimal infrastructure. In *USENIX Conference Proceedings*, pages 131–138, San Diego, CA, June 2000. USC/Information Sciences Institute, USENIX.

[9] A. Murugappan and L. Liu. An energy-efficient approach to processing spatial alarms on mobile clients - git-cercs-08-03. 2008.

[10] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Trans. Comput.*, 51(10):1124–1140, 2002.