Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 5, May -2017

The Review Friend Tracing-Effective Vicinity Detection between Mobile Crony Groups

More Rupali Kalu[1], Deshmukh Sakshi Vilas[2], Shaikh Zeba Akhlakh[3]
Nikam Sayali Dnyaneshwar[4], Prof. Dhande M. T[5]

[1,2,3,4] *B.E.Computert, Computer Department, Shatabdi Institute of Engineering & Research, Nashik*
[5]*Assistant Professor, Computer Department, Shatabdi Institute of Engineering & Research, Nashik*

Abstract —*Worldwide positioning systems (GPS) and cellular phone networks are making it feasible to song man or woman users with an increasing accuracy. Its miles herbal to invite whether or not you can still use this information to preserve social networks. Here every person desires to be knowledgeable on every occasion one in all a list of other users, called the person's pals, seems in the user's vicinity. In evaluation to more conventional positioning based algorithms, the computation right here depends no longer only on the consumer's very own position on a static map, but additionally at the dynamic function of the consumer's friends. Hence it requires both verbal exchange and computation sources. The computation can be executed both between the person users in a peer-to-peer fashion or via centralized servers in which computation and facts may be collected at one primary area. In the peer-to peer model, a novel algorithm for minimizing the wide variety of location replace messages among pairs of buddies is provided. We additionally gift an efficient set of rules for the centralized model, based on region hierarchy and quad trees. The paper presents an evaluation of the 2 algorithms, compares them with a naive technique, and evaluates them the use of the ibm metropolis simulator machine.*

Keywords- *strips set of rules, area based services, social networks, worldwide positioning systems, dynamic nearest pals*

I. INTRODUCTION

Global positioning structures and cell cellphone networks make it feasible to track character customers with a growing accuracy. One appealing software of knowing the geographic region of customers is to compute and maintain social networks. In those networks, each consumer can also specify or be related to a set of other customers, known as the person's buddies. Whenever a chum movements into the user's vicinity, each customers are notified with the aid of a proximity alert message. In an extra well-known context, a social institution is one that is predefined via enrollment or through matching the non-public profiles of customers. A set may also seek advice from a listing of people however also to different businesses of individuals. We use the time period region to consult a place round the user. On this paper, an area is represented via a circle of a pre-specified radius, which can be uniform for all users, or defined for each pair of buddies. The proposed set of rules for the peer-to-peer model can certainly accommodate a specific vicinity radius for each pair of pals, in addition to other convex vicinities. Different definitions of area, and even dynamically changing definitions, are possible. As an instance, the radius may trade between daylight hours and night time, it'd rely upon the user's area, and it is probably a non-circular form. Global positioning systems and cellular telephone networks make it feasible to track character customers with a growing accuracy. One appealing software of understanding the geographic location of clients is to compute and preserve social networks. In those networks, every client also can specify or be associated with a hard and fast of different clients, referred to as the person's pals. Every time a chum moves into the consumer's vicinity, every customers are notified with the aid of a proximity alert message. In a greater widespread context, a social organization is one that is predefined thru enrollment or through matching the private profiles of customers. A hard and fast may additionally are trying to find advice from a list of humans however additionally to one of a kind corporations of people. We use the term location to consult a place round the consumer. On this paper, a vicinity is represented thru a circle of a pre-specified radius, which may be uniform for all users, or described for each pair of friends. The proposed set of policies for the peer-to-peer version can actually accommodate a specific area radius for each pair of buddies, similarly to different convex vicinities. Different definitions of region, and even dynamically changing definitions, are viable. For instance, the radius might also alternate between daylight and night time, it would depend upon the consumer's place, and it might be a non-circular shape. Entities, or users, as well as a repetitive computation of all pal's distances after each such location update. This will be a very inefficient process. Preserving social networks based totally on consumer places is an exciting problem from the element of computational geometry and from a database perspective. It's also thrilling from the factor of view of a dispensed system; the method is computationally pricey, but there may be an efficient manner to cut up the computational assignment among distinctive geographic places. We distinguish between two extraordinary computational frameworks. In the centralized computation model, customers ship their location facts to a centralized server which continues monitoring of person locations and lists of friends and is accountable for computing and sending the alert messages to all pairs of buddies. The second, peer-topeer computation model, involves no crucial server. As an alternative, every pair of pals is responsible for maintaining each different informed about their area, detecting place events, and transmitting alert messages.

II. SCOPE AND OBJECTIVE

2.1. Scope

We use the term vicinity to refer to an area around the user. In this paper, a vicinity is represented by a circle of a prespecified area, which can be constant for all users, or defined for each pair of friends. The suggested algorithm for the peer-to-peer model can naturally provide accommodations a different vicinity area for each pair of friends, as well as other rounded vicinities. Other definitions of vicinity, and even dynamically changing definitions, are possible. For example, the range might change between daytime and night time, it might depend on the user's location, and it force to be a non-circular shape. We differentiate between two different computational contexts. In the centralizedcomputation model, users send their location information to a centralized server which cling to tracking of user locations and lists of friends and is liable for computing and sending the alert messages to all pairs of friends. The second, peer-to-peercomputation model, includes no central server. Instead, each pair of friends is liable for keeping each other informed about their location, identifying vicinity events, and transmitting alert messages .The estimation of an algorithm for keeping social networks is an important problem. The number of messages would depend not only on the number of users, and the distances between them, the vicinity are and the preferred acceptance, but also on the nature of their signal routes and relations between them. In the computational geometry fiction, the dynamic model is a common model for evaluating the effectiveness of algorithms for keeping dynamic structures. In this model, the role of the evaluated algorithm is to maintain some symmetrical properties for sets of moving elements, where each element moves along a low-degree statistical curve. From time to time, an eventoccurs, in which new elements may be inserted and existing elements may be deleted or may change their routes. The number of changes in the data structure is estimated as a function of the number of events in the active input data set.

2.2. Objectives

Algorithms for monitoring moving gadgets are determined in mobile computing literature, both inside the database community, and inside the mobile communications community. Lots of the work assumes that shifting objects are represented with the aid of easy factor items whose places are continuously updated in an index. This however calls for non-stop updating of the locations of all users, which might result in a large wide variety of location messages. Trajectory-based totally algorithms are becoming increasingly more popular. Storing and indexing trajectories enables now not only green spatial variety queries but time-and-space range queries. See additionally. Discusses time-parameterized bounding rectangles and extends trajectory facts with expiration facts.
There may be a big body of literature on retaining a particular property of transferring items. For example, a randomized algorithm for preserving the binary space partition of moving gadgets is mentioned in, and the renovation of the dynamic coronoid diagram of a fixed of shifting points in an aircraft is presented in. For preserving and querying a database of moving objects. Diverse algorithms have been supplied for indexing transferring points. Their foremost concept is to apply a linear function of time for each of the dynamic attributes of the item, and to provide strategies to regenerate the quad tree. Their algorithm presents indexing of gadgets transferring in 1, 2 or three dimensions. For different paintings in query processing for transferring factors, which proposes algorithms for variety question and ok nearest neighbors. Even as the dynamic facts structures or databases mentioned above can be green for other styles of queries, the mission in hand isn't correctly treated by means of any of them. As an instance, some of those facts structures might be used for querying at a particular time example who're the pals in the vicinity of a single person, round its present location. But, consider that these days there are loads of tens of millions of cellular cellphone users within the international. Again and again tracking all customers and querying their vicinities in the sort of huge populace requires a massive wide variety of messages to be exchanged and a number of computation and could be very inefficient. To the satisfactory of our know-how, the trouble of keeping social networks has not been addressed earlier than. The algorithms and information structures proposed in this paper are designed to correctly cope with this mission. However they won't be as efficient for different, conventional types of spatial queries, for that reason being complementary to the above preceding paintings. This work is also complementary to the hassle of finding human beings whose non-public profiles healthy. For this problem, industrial solutions have been provided.

III. LITERATURE SURVEY

In cellular networks (e.g. mobile phone networks), a fractional approach is to try to make benefit of the usual cells structure enforced by the network. If A is around the area of a cell, then one requirements to keep track of friends enumerated to the user's own cell and nearby cells. However, in general this method might be insufficient because the cell sizes fluctuate greatly, ranging from large macro cells in rural areas to tiny Pico cells in urban areas and buildings. Different users might as well define different locality radii for different friends, and these force even variation when they move from one place to another. For example, a marketing manager does not want to be alerted in his office of all his contemporaries who are close by in the office, but may want such alerts when the same contemporaries are within a city

block distance on an overseas trip, as this is a chance encounter. Also note that not all wireless communication is based on cellular networks in the first place.

IV. EXISTING SYSTEM

Evaluating between these two algorithms is not an obvious project. The Strips set of rules, designed for peer-to peer operation, goals at minimizing the communication complexity, namely the number of region update messages being dispatched between pairs of users. Alternatively, the centralized, quad tree-based set of rules, targets At minimizing the computational complexity, assuming that it knows wherein are all the customers at all times (or, at least in any respect mobile crossing activities). Also observe that the Strips set of rules contains any values of R and, whilst the quad tree algorithm is constraint to a uniform R price and a single, rough tolerance Criteria, = (221) those difference limit the ability to compare between the two algorithms. We evaluate between the two by using counting simple operations in both, as a result blending among computational and verbal exchange complexities for the sake of plotting one combined graph. The wide variety of simple operations relative to the number of friends per consumer, for the stepped forward quad tree method, naive quad tree technique and Strips technique. A basic operation in the Strips set of rules is a location update message, which correspond also to 1 strip replace. A basic operation inside the centralized algorithm is checking whether a selected pal of the user is in any of the close by cells. Accordingly each time a consumer crosses a cellular boundary, the number of fundamental operations executed by using his cell phone is equal to the minimum among the wide variety of buddies and the wide variety of acquaintances the user has. The graph suggests that the range of primary operations is about linear with the number of friends in all the three instances. Word however that, relying on the range of customers inside the neighboring cells, the quantity of operations may want to grow slower than linear. Additionally note that the variety of simple operations in the strips set of rules is a whole lot decrease than the quad tree based algorithms. Its mile sob vinous from hat figure that Strips algorithm plays higher than the improved quad tree method which in turn plays higher than the naïve quad tree approach. If we put the conversation thing apart, it way that the Strips set of rules also computes much less times the distances among user pairs. Hence when implemented on a centralized server, the Strips set of rules can be more efficient than the quad tree set of rules. Accordingly we see that the Strips set of rules outperforms the quad tree set of rules in both centralized and distributed settings.

V. PROPOSED SYSTEM

A. Mobile client

The cellular patron consists of a cell cellphone and a GPS receiver which can be used to find the region of own family and pals. The cell client can ship a pop up SMS about the area to the person while someone is close.

B. Repository

The repository includes all the records approximately the users, region maps, and the vicinity-associated outcomes.

C. Web client

The records within the repository can be managed and considered the use of the internet client. The person receives the location information from the net patron on their mobiles.

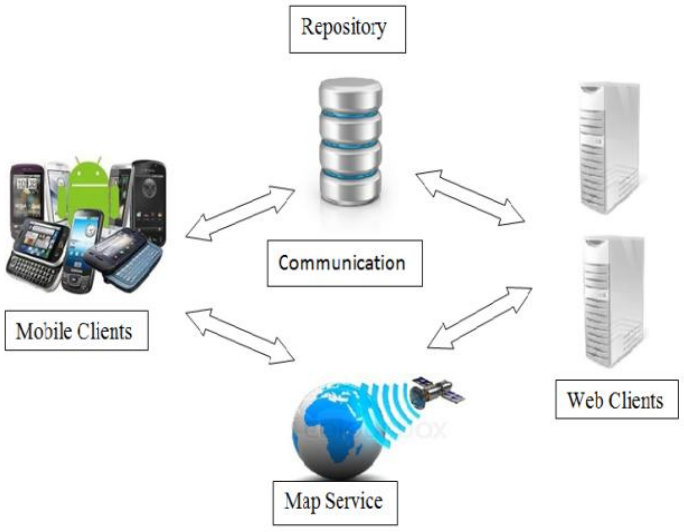

Fig 1: Proposed System Architecture

D. Map service

The map provider is an agent primarily based which provides both the mobile and the internet customer with map facts. The map provider uses GPS to tune the position of friends or family members. The region statistics is up to date to web patron every time through the cell smartphone.

E. Message Alert system

The message alert system compacts with perceiving location of our buddy and member of the family and bring up to date on server. It sends position bring updated to the consumer whilst buddies are within specific range from him/her.

VI. PROPOSED ARCHITECTURE

A. Login Module

Login module is used to offer registration of recent person and log into the machine. Check in interface takes user records and after the registration is a hit, person can login to the device.

B. GPS Module

GPS module offers with role based services. The usage of this module, user can discover his/her own function, buddy function and family member role.

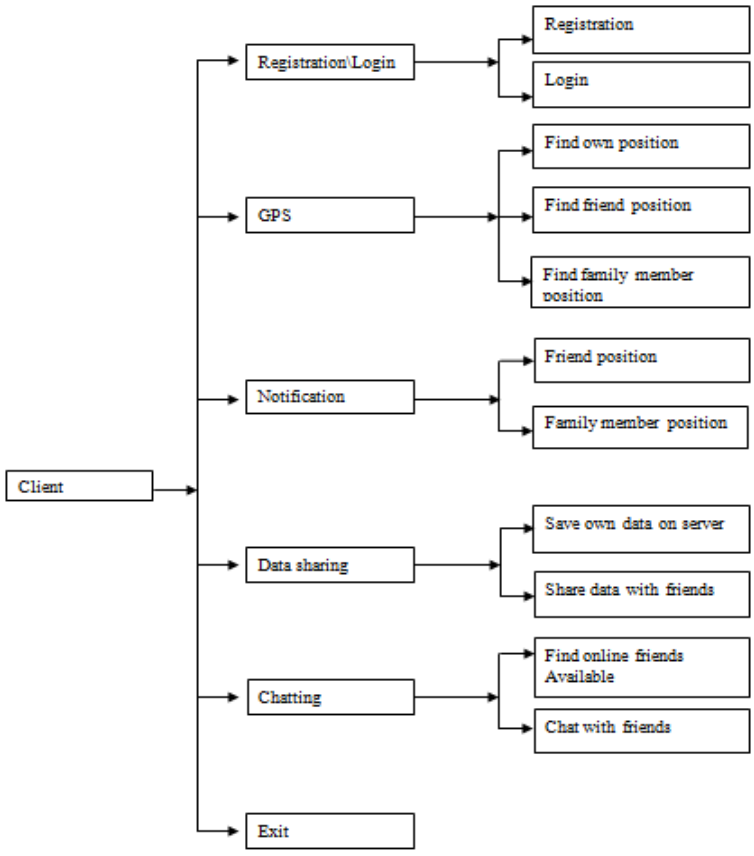

Fig 2: Proposed Architecture

C. Data sharing Module

Facts sharing module offers with sharing the textual content facts.

D. Notification Module

Notification module sends a notification to the consumer whilst his/her pals or family members come across the user's vicinity of path. A notification is given to the user inside the system of a popup message having the position records alongside mild or sound.

E. Chatting Module

Chatting module is basically to enforce word communication amongst pals. For chatting, user can take a look at for the friends handy online.

VII. RESULT

We present a novel distributed algorithm, denoted as the strips algorithm, in which pair of moving friend's make an agreement about a static buffer region between them. After the agreement is made, they do not need to track each other's location until one of them enters the buffer region for the first time. By doing so, the agreement is terminated. Hence they exchange a location update message between them, check if they are within the R-vicinity of each other, and otherwise make a new agreement on a new static buffer region. We provide an exact and an approximate strip's algorithms, supported by analytical and empirical results that show their efficiency. When analyzing an algorithm for such a problem, one has to consider both communication and computation complexity. For this distributed algorithm we focus on reducing the communication complexity, or the required "air time", which has a significant impact on the battery lifetime of a mobile communication device. It is shown that the number of messages is logarithmic in the distance between the users when they start to approach each other from far away. It is also logarithmic in 1/ε when they are getting closer, where ε is the desired tolerance for producing the proximity alert. Hence we consider it to be a very efficient algorithm. A second, quadtree based algorithm is presented for the centralized approach. This algorithm aims at reducing the computational cost, assuming that all users periodically update the server with their location as they move. Somewhat surprisingly, we found that the quadtree based centralized algorithm is inferior to a centralized implementation of the strips algorithm on such a central server.

A. Proposed System Overview:

In this distributed, peer-to-peer model, it is assumed that each user carries a wireless device that knows its own location and has enough computational power for a local computation. In order to compute its distance from a friend it needs to get the location of that friend, and this requires a location update message to be sent. Our objective is to minimize the communication complexity, or the number of location update messages exchanged with devices of other users. Let a, b be two users whose Euclidean distance, denoted |b−a|, is larger than denote the bisector of the line connecting a and b; i.e., the line consisting of all points of equal distance from a and b .

B. The data structure D(a) for the Strips algorithm:

```
SelfMotion() {
   do // repeat while moving
      a=ReadSelfLocation()
      Test(D(a))
         if (a enters S(a,b_i),
             (for some i))
              or MsgReceived(b_i))
         StripUpdate(b_i)
   enddo }
```

```
StripUpdate(b_i) {
   send a's location to b_i.
   receive b_i's location.
   if |a − b_i| < R + ε
      ProximityAlert("b_i is nearby")
      Delete( D(a), S(a,b_i))
   else
      Compute S(a,b_i)
      Update D(a), D(b_i) with S(a,b_i)
   end
}
```

Fig 3:Strips Algorithms

The selection of ε determines a tradeoff between the desired distance accuracy in generating alerts and the required number of location update messages. First we address an obvious, yet an important stability aspect of the algorithm, which depends on ε.

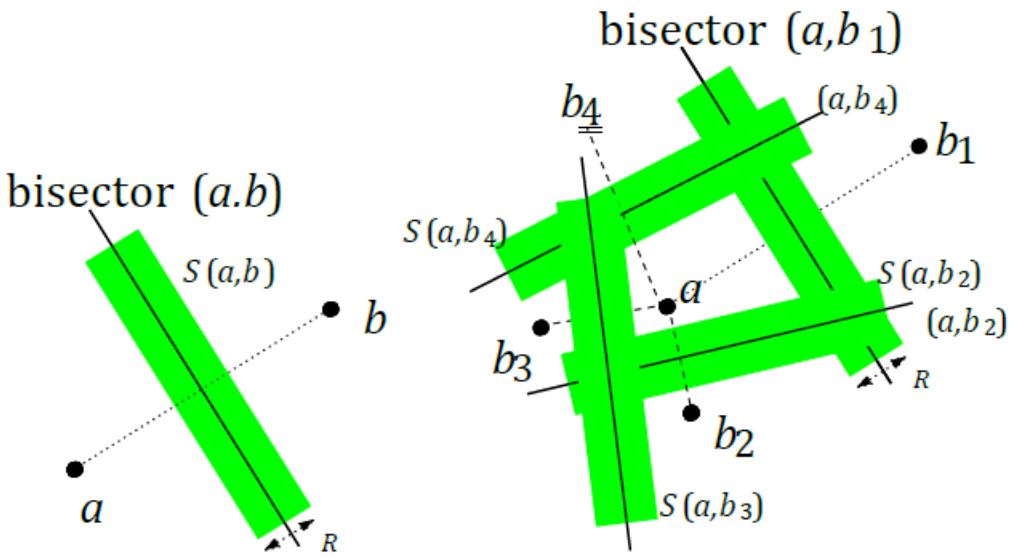

Fig 4: Left: setting a new static strip of width R around the bisector between two mobile users. Right: user a does not need to update strips while moving inside the internal region (P).

COROLLARY 3.1. Two users have to move a total distance of at least 2ε between any two proximity alerts they generate. In order to generate two proximity alerts, one user has to enter the vicinity of the other, then exit the vicinity, and then enter it again. Hence the users are at most R + ε apart when the first proximity alert is invoked, then they are at least R+2ε apart when they get apart, and then at most R+ε apart when the second proximity alert is invoked. This corollary ensures that the algorithm state will not change back and forth in infinitely small time periods when the two users are moving around the boundary of the vicinity region.

In order to generate two proximity alerts, one user has to enter the vicinity of the other, then exit the vicinity, and then enter it again. Hence the users are at most $R + \varepsilon$ apart when the first proximity alert is invoked, then they are at least $R+2\varepsilon$ apart when they get apart, and then at most $R+\varepsilon$ apart when the second proximity alert is invoked. This corollary ensures that the algorithm state will not change back and forth in infinitely small time periods when the two users are moving around the boundary of the vicinity region.Next we illustrate the role of ε in the algorithm termination criteria (i.e., announcing a proximity alert). Consider a simple case of two users a and b. Let user a be stationary, and let user b be moving on a straight line towards user a.

Denote the initial distance between a and b by R + x, for a positive x. When user b hits the strip, its distance from user a would be [(x+R/2) + (R/2)=(x/2)+R] half the initial distance x plus the width of the strip, R). Similarly, the next strip will be located such that will hit its boundary at a distance from user a of Hence it is clear that this sequence of strip-update events forms a series of distances which is the sum of one constant component, R, and a geometric series,

COROLLARY 3.2. The number k of strip-update events performed by a stationary user a and a user b moving on a straight line towards a from a distance the number of messages exchanged between users a and b is logarithmic with the initial distance between them, and is also logarithmic with 1/ε, the inverse desired tolerance.

This reflects the trade off between the desired accuracy and the required number of location update messages. It is a very small number of updates, demonstrating the efficiency of the algorithm. As indicated earlier, as ε decreases to zero, the number of messages k increases to infinity. That is, an alert at the exact time would require an infinitely large number of location update messages. By introducing a tolerance ε > 0 into the model we avoid this undesired case.

C. The general case of moving on a straight line :

Next we consider a more general case in which b is moving on a straight line, but the stationary user a is located at distance d from the line. Figure 3 shows a typical sequence of updates. The point b is moving from right to left along the y = 0 line, and its positions at the times of hitting the strips are marked by small circles, labeled with the update serial number. The strips are shown in dashed lines, and are similarly numbered. For instance, when user b hits strip number 1 it defines the location of point 2, and so on. In this example b passes nearby a but out of its vicinity and no proximity alert is produced. After the 8th update b would continue to move and no more updates will be done.

The messages is required when b passes very close to a, but still keeps out of its R-vicinity. The region in the graph for which d < R terminates with a proximity alert after producing a number of strings which is linear with log(1/ε) (note the logarithmic axis, which represents ε values between 0.00001 and 10.0). The region in the graph for which d > R corresponds to cases where there is no proximity alert, and thus after getting away from the transition area along d = R, the number of messages does not depend on ε at all.

Hence we see that for long distances the number of messages is logarithmic with the distance, and for short distances the number of messages is logarithmic with 1/ε. We consider it to be a very efficient property of the algorithm. In this section we provide analytical and numerical analysis of basic cases to illustrate the efficiency of the strips algorithm.

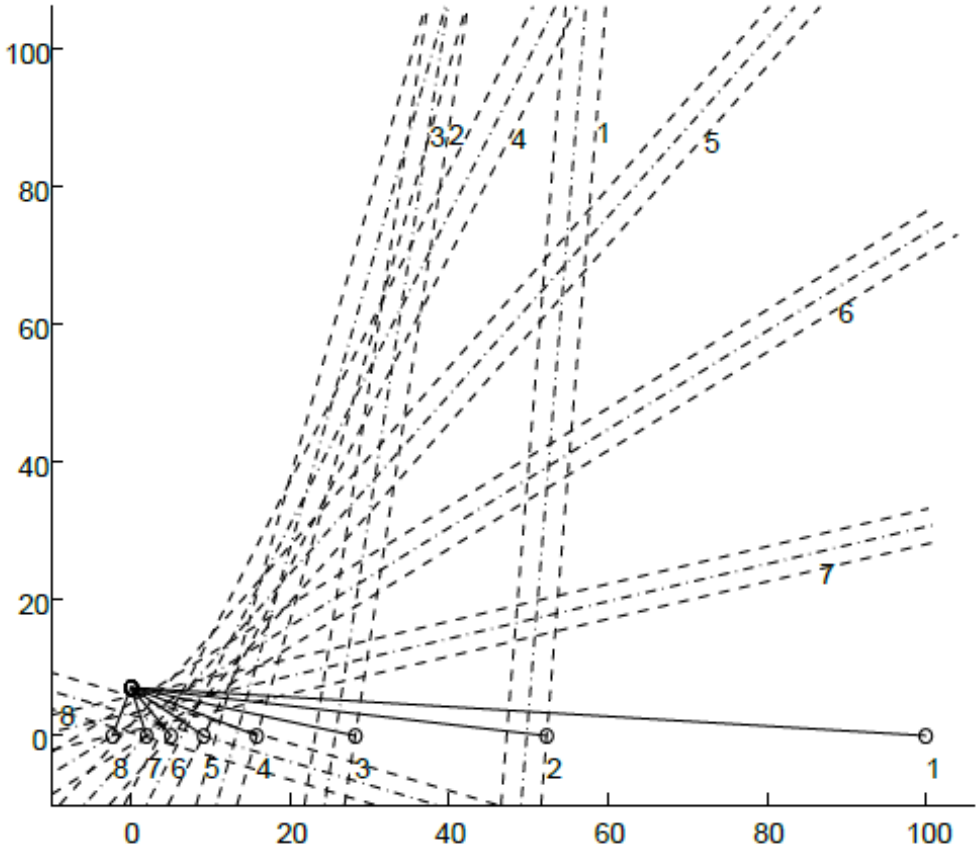

Fig 5: A sample series of strips for a user b moving on the x axis from right to left, where user a is at (0,d).

We further analyze this behavior for the case of ε = R, which is later used in some of the simulations. Let a and b be two users, with a staying at a fixed position and b moving in a straight, vertical trajectory towards a point c. Let d denote the horizontal distance of b from a, and let v be the vertical distance of b from a. Assume also for simplicity that users exchange location update messages when they hit the center of the strip. Theorem 3.1 upper bounds the number of location update messages sent.

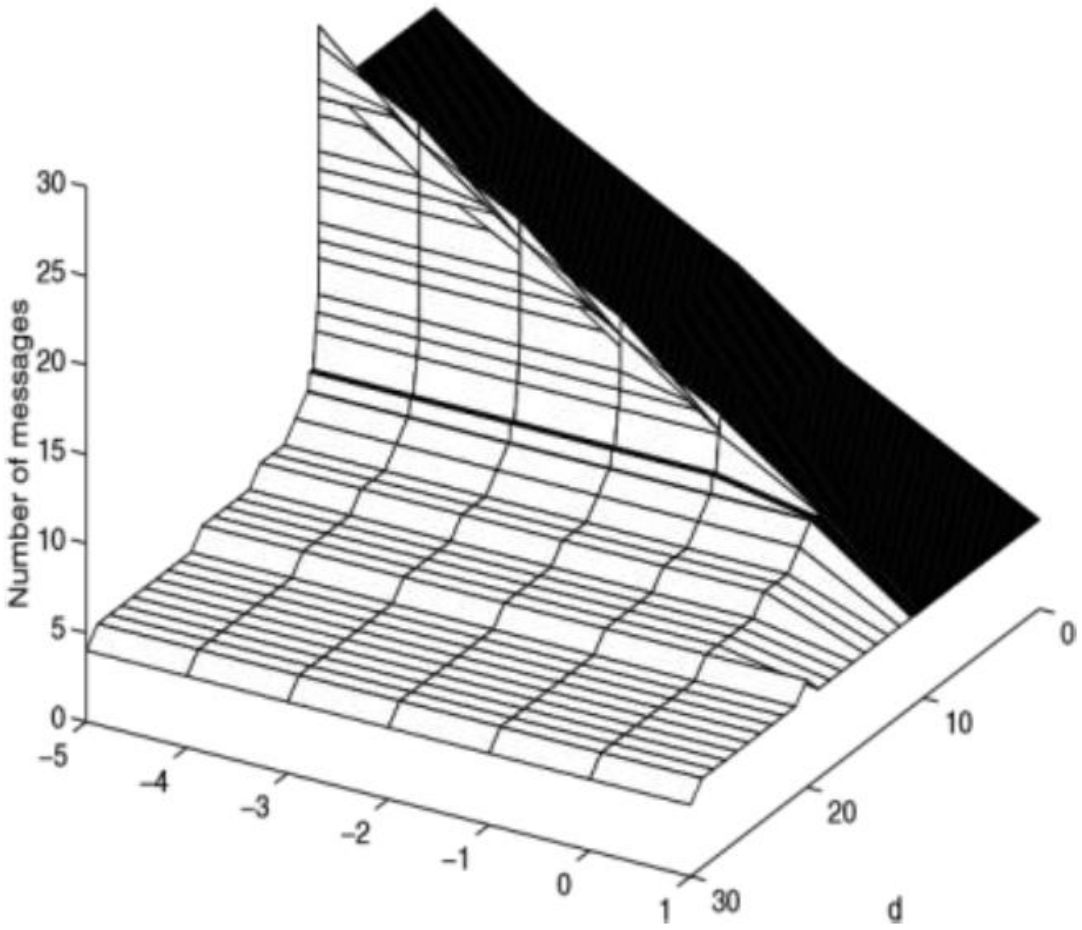

Fig 6: The effect of d and ε on total number of strip updates (R = 10.0, ε is given in a logarithmic scale). Two regions are observed, one corresponds to d < R and the other to d > R. See text for details.

The number of times that the strip is updated is $\leq \log(v/(d - R))$ if $d > R$, and is $\leq \log(v/R)$ if $d < R$.
Users who move on arbitrary paths may cause an increasing number of strip updates over time. In the worst case, a user has to move at least $\varepsilon/2$ between two location update messages. This is a tight bound, achieved when one user follows a friend on a straight line, at the same speed, and with a distance slightly larger than $R + \varepsilon$.

D. The "naive" quadtree algorithm :

Let Γ be a partition of the plane into *regions*, defined recursively as follow
• Initially Γ consists of a single square region, covering the entire area of operation
• Let c be a region of Γ. If c either contains ≤ 1 users, or its edge-length is $\leq R$, stop.
Otherwise, replace c with 4 equal-size squares $R1, R2, R3, R4$, representing its four quadrants. This division imposes a quad-tree data structure T, with the property that every leaf region whose size is larger than R contains at most one user. We call a region containing more than a single user a *live region*. Hence live regions are always of same edge size, $\leq R$, and from here on we assume their size is exactly R. We augment T so that it is a *balanced* and a *netted* quad-tree. That is, the difference in size between two neighboring leaf regions is at most a factor of 2, and each leaf region maintains pointers to all of its neighbouring regions.For every user *bi* let $F(bi)$ denote the list of friends of *bi*. For every region $c \in \Gamma$ let $U(c)$ denote the users currently inside c. The basic idea is as follows. Once user *bi* registers with the system and reports its location, the system seeks friends of *bi* in the region $c \in \Gamma$ containing *bi*, and in the neighbouring regions of c. Note that there are at most 8 neighbouring cells of c, since T is a balanced quad-tree. When *bi* moves from region c to a new region c, we only need to check if any friend(s) of *bi* are found in any of the *new* neighbouring cells. We next explain each stage in detail.

E. Finding the region *c* containing *bi*.

Let *h* denote the height of the quad-tree T . A simple approach would be to use T itself for this point location task. It requires tracing the path of length ≤ *h* from the root of T and to the leaf associated with *c*. Alternatively, we can use the point location technique of [4] that requires only *O*(loglog*h*) time. However, in many cases, we can do better; Note that the size of all live regions containing two or more users is exactly *R*. The coordinates of the left lower corner of such cells is (*mxR*,*myR*), where *mx*,*my* are integers. We refer to the pair (*mx*,*my*) as the *index of c*. Note that for every point $p = (x,y)$, the index of the live leaf region containing *p*, if one exists. Hence we store all the live regions in a hash table where the key is the index of the live region. We maintain a pointer from the hash cell to the leaf of T associated with this region. Hence finding users within the same region as user *p* is done in expected time *O*(1). Once this region is found, then by using the properties of the netted quadtree, finding the neighbour regions is done by following the links to the neighbours, in *O*(1) worst-case time.

F. Finding if there are any friend of *bi* in its vicinity.

For each live region *c* we maintain a hash table of all the users currently inside *c*. We also maintain for every user *bi* a hash table of its friends. When *bi* enters a new region *c*, we compare the length of the friends list with the length of the combined lists of occupants in the (up to four) neighbouring regions to *c* which are not neighbours of *c*. If the friends list is shorter, we check the distance to every friend of *bi*. Otherwise, we check for all occupants of the neighbouring regions which of them is a friend of *bi*. Thus the running time is $O(\min\{|F(bi)|,|U(c)|\})$ (in the expected sense, due to the use of hash tables).

G. Improved centralized quadtree algorithm:

The purpose of the improved algorithm is to reduce the size of the list of friends that *bi* needs to check upon entering a cell *c*. Let $q1$,$q2$ and $\pi(q1,q2)$ denote two nodes in *T* and the path in T from $q1$ to $q2$, respectively. Let *p*, *Rp* denote a node in the tree and the region in the plane associated with it, respectively. We define the lists *Fp*(*ci*) in a bottom up fashion. If *p* is the leaf node, we define *Fp*(*bi*) to denote the set friends of *bi* which are in *Rp*. If *p* is not a leaf node, we define *Fp*(*bi*) to denote the friends of *bi*, which are not *Fp'* in for any decedent node *p' of p*. In other words $bi \in Fp(a)$ if and only if *p* is the lowest common ancestor of the leaves nodes containing *a* and *bi*. An intuitive way to understand this definition is to think of *Fq*(*bi*) as all the friends of *bi* who are in the same city as *bi* but are not in the neighbourhood of *bi*.**Entering a new user.** When a new user bi registers into the system (e.g., by turning on its cellular phone), we find the leaf region Rp containing bi, check the list F(bi) of bi's friends, check their location in T , and create the lists Fp(bi). This can be done efficiently in expected time O(h + |F(bi)|), where h is the height of T .While this data structure is more complicate than the naive quad-tree, it allows us to derive some theoretical bounds on the number of updates which will occur for certain families of motion trajectories. These support our claim that this data structure is efficient.*Let γ be a curve in the plane, with the property that every straight line crosses γ at at most K points, where K is a small constant. Let bi move along γ, and assume that all its friends are stationary. Then the total number of checks and updates done by the improved centralized algorithm is only O*(*nih*), *where ni is the number of friends of bi, and h is the height of* T . *This bound is tight in the worst case.*Shows that for complexity-bounded motion trajectories the number of updates is also bounded (proof is omitted). It is harder to give any complexity bounds if all friends of *bi* are allowed to move arbitrarily. For example, it seems that if all of them move together with *bi*, then there is nothing we can do in this algorithm except checking all of them each time *bi* enters a new cell. However, if most of them are located far away from *bi*, for most of the time, then we expect this algorithm to be quit efficient. The number of messages sent by the exact Strips algorithm and by the approximated Strips algorithm with bounding rectangles.

H. Experimental Results :

We show the number of proximity alert messages, plotted against R, the radius of user's vicinity. Note that this simulation is independent of the algorithm used. It can be seen that the number of alerts reaches a maximum at a midpoint, for R = 275. This is expected, as if R is very small then getting two friends close enough has very small chance. On the other hand, if R is very large, then nearly everyone are in the vicinity of all their friends, and thus very few new meeting events are likely to occur. In between, for mid range R values, there is a higher chance for a pair of users to alternately get close enough to each other and then get apart from each other. This might occur several times while they move along their traces. decreases as R increases, because the minimal cell size is also R. Note that this number represents the number of times the algorithm needs to check the neighboring cells for friends of the user who crosses cells.

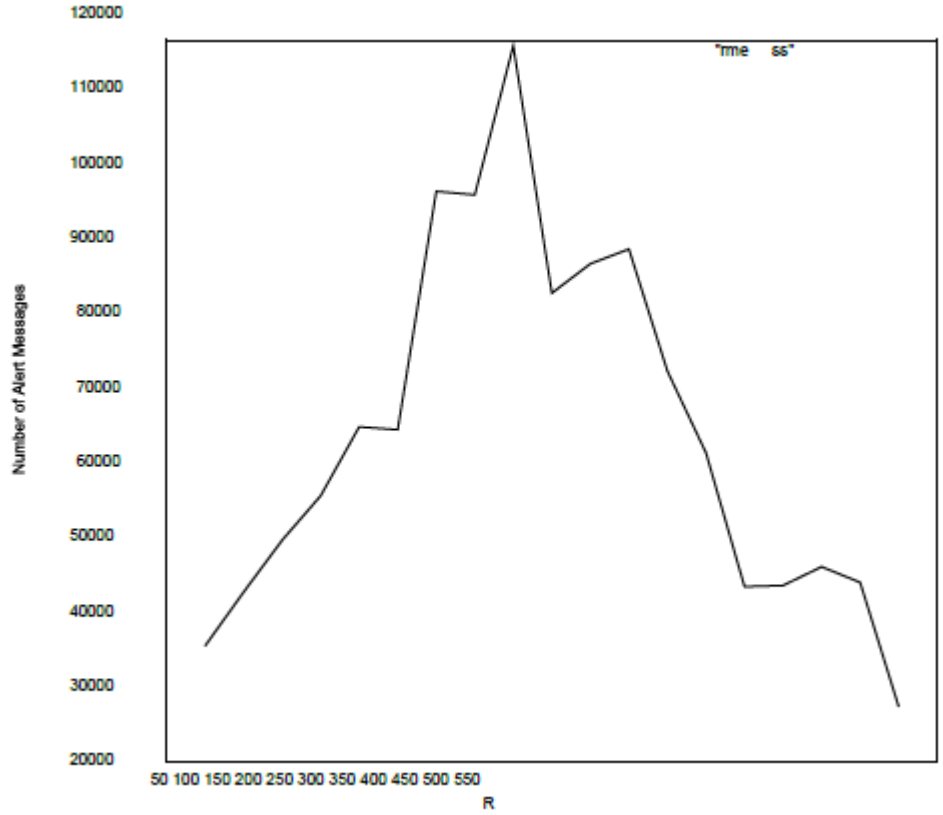

Fig 7: Number of Proximity Alert Messages Generated as a Function of R

The number of strip updates, each correspond to one location update message in the Strips algorithm. In general, increasing R causes less number of strip updates to occur. It is analogue to shortening the traces, as the actual length of the trace should be measured with respect to ε, here equal to R. Although it is somewhat similar to the result for the quad-tree algorithm, note that here we count all the communications between pairs, while in the previous graph we count only the individual's events of passing from cell to cell. One depends on the number of friends, while the other does not.

REFERENCES

[1] P. Agarwal and C. M. Procopius, "Advances in indexing mobile objects", IEEE Bulletin of Data Engineering, to appear.
[2] P. K. Agarwal, L. Arge, and J. Erickson. "Indexing moving points. In Pro c. of the ACM Symposium on Principles of Database Systems (PODS)", 2000, 175-186.
[3] P. K. Agarwal, J. Erickson, L.J. Guibas, "Kinetic Binary Space Partitions for Intersecting Segments and Disjoint Triangles", Pro c. Ninth Symposium on Discrete Algorithms, 1998,107-116.
[4] D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, "On Dynamic Voronoi Diagrams and the Minimum Hausdorff Distance for Point Sets under Euclidean Motion in the Plane," in Symposium on Computational Geometry, 1992, 110-119.
[5] H. D. Chon, D. Agrawal and A. E. Abbadi, "Range and kNN Query Processing for Moving Objects in Grid Model," Mobile network and Applications, To Appear.

