Proactive Neighbor Localization Based on Distributed Geographic Table

Marco Picone, Michele Amoretti, Francesco Zanichelli
Dept. of Information Engineering
University of Parma
Via Usberti 181/a
43124 Parma, Italy
{picone,amoretti,zanichelli}@ce.unipr.it

ABSTRACT

Real-time tracking of massive numbers of mobile devices, either carried by humans or embedded into vehicles, is a challenging problem whose solution may pave the way for a large set of valuable applications, ranging from social networking to ambient intelligence. A centralized approach, i.e. a server collects position data and provides it to interested consumers, is highly questionable, as performance can hardly scale up to the needs several million concurrent users. On other hand, a decentralized peer-to-peer approach, for which positioning data would flow directly among mobile devices may be very appealing , provided that messages to be routed are not too frequent and too expensive in terms of bandwidth usage. In this context we propose a peer-to-peer overlay scheme called Distributed Geographic Table (DGT), where each participant can efficiently retrieve node or resource information (data or services) located near any chosen geographic position. In particular, we describe a DGT-based localization protocol, that allows each peer for proactively discovering and tracking all the peers that are geographically near to itself. We provide a performance analysis of our protocol, referring to a simulated (although realistic) scenario where several hundred vehicles move on a real map. Our results show that the solution is efficient, scalable and highly adaptable to different application scenarios.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.4 [**Distributed Systems**]: Distributed applications

Keywords

peer-to-peer,neighbor position discovery,localization,mobile computing

MoMM2010, 8-10 November, 2010 Paris, France

1. INTRODUCTION

Real-time localization services are gaining more and more importance for a broad range of applications, such as road /highway monitoring, emergency management, social networking, and advertising. For example, Google Latitude [5] detects the user's location (using Wi-Fi, 2G/3G/4G mobile or GPS satellite signals), and allows for sharing it with authorized contacts, thus anyone can be aware of the location of his/her friends. In such a system, however, two major drawbacks can be found. The first one is technical, *i.e.* all information is managed by centralized servers, that makes the service not massively scalable (unless you have the computational power of Google). The second one is more "social": the service does not enable users to select a region and discover all the people that are located there, thus preventing for example the establishment of new social relationships or the local dissemination of service advertisements.

Granted that users have to agree on allowing their localization and tracking, we foresee location-based services (LBS) supported by an infrastructure implemented as a network of distributed software entities, with flat or hierarchical organization, but always without relying on a central entity. As an example of such LBSs, let us consider a traffic monitoring application spread over a set of nodes placed along highways, each one collecting information about local traffic, and being able to provide aggregated information (*e.g.* statistics) to any remote user that requests it.

Any centralized approach would hardly scale for such highly variable contexts, with mobile nodes changing their location very quickly, in a wide and highly populated area. Indeed, search results would be incomplete or outdated with high probability. Like other authors (see the Related Work section) we believe that a partially or fully decentralized approach can increase the accuracy of information and the rate at which they are retrieved by users. Moreover, it may allow to update and publish information directly, with low cost and high scalability. Last but not least, it may simplify the process of joining the virtual community and publish new services.

In this paper we describe a general framework called Distributed Geographic Table (DGT), that defines a peer-to-peer strategy for mobile node localization, and a particular instance that supports applications in which every node requires to be constantly updated about its neighbors. Compared to a centralized approach, DGT is more scalable, since its performance (in terms of responsiveness, completeness and robustness) remains valuable also for a large number of

nodes and when the nodes' dynamics are very high.

The rest of the paper is organized as follows. Section 2 presents a survey and discussion of state-of-art research on location based search in P2P networks and underlying space representation techniques. Section 3 introduces the main concepts of Distributed Geographic T able, including a formal definition of *neighborhood*, as well as the requirements for its routing and maintenance strategies. Section 4 describes the idea and the architecture and an overview of different procedures used to maintain the overlay. Section 5 introduces our simulation results with three different scenarios used to show the behaviours with different parameter's configuration. Finally, we conclude with a discussion of our contributions and with some ideas for future work.

2. RELATED WORK

Community oriented architectures for geographic based services, often referred to as *geocollaboration* frameworks. The need for collaborative work with geospatial data has escalated in recent times also due to events such as terrorist activities and natural disasters. Distributed localization, a clear example of geocollaboration service, is usually implemented by recursively dividing the 2D space into smaller areas in order to assign responsibilities for region of space to peers. Instead of employing a number of centralized servers (either dedicated or selected among participating nodes) to carry the load for the entire network, every node in the network shares the load of indexing and searching data that refers to its area. The idea of hierarchical partitioning comes from the indexing of data structures for multidimensional data-sets such as R-tree [12] that is widely used in centralized databases. An overlay structure allows for routing queries within the system. This means that each time it is necessary to know which peers are located in a certain area, a number of lookup queries must be sent.

Examples of general-purpose peer-to-peer hierarchical schemes are HZSearch [8], DPTree [7], DiST [9]. These and other works [10], [11], [3] propose strategies for supporting complex queries over multi-dimensional data, such as "select five available buildings closest to the airport". The specific problem of geographic localization is addressed by Globase.KOM (Geographical LOcation BAsed SEarch) [1], which adopts a tree-based P2P overlay enhanced with interconnections. Globase.KOM uses the more powerful nodes with good network connectivity, which tend to stay online for a long time as supernodes in Globase.KOM. Supernodes are responsible for indexing all nodes/services in one clearly defined geographical area. The "normal" (non-super) nodes in the network simply offer and consume services without having additional responsibilities. The idea is that the world projection is divided into disjoint, non-overlapping zones. Each zone is assigned to a supernode located inside the zone which keeps overlay/underlay contact addresses for all nodes in that zone. Supernodes form a tree where node A is called the parent of node B when B's zone is inside A's zone. Another architecture, called GeoP2P [2], still performs a hierarchical partitioning of the 2D geographic space, but adopts a fully decentralized peer-to-peer overlay scheme, with overlay maintenance and query routing performed without super or special peers.

The main drawback of the hierarchical approach is that peers representing higher level zones may become bottlenecks for query routing, and possible points of failure for the whole system. Moreover, none of the state-of-art solutions has been demonstrated to work in presence of mobile peers. Our DGT framework takes into account mobile nodes, enabling *disruption-tolerant networks* (DTNs) [13]. DGT is based on the principle that different geospatial applications have different information needs, for which it is not fair to constrain the framework with a unique data management strategy. For example, our implementation described in section 4 adopts a proactive data collection strategy that makes it highly suitable to target those situations where each user must be always aware of which nodes are in his/her surroundings.

3. DISTRIBUTED GEOGRAPHIC TABLE

By Distributed Geographic Table (DGT) we refer to an overlay scheme where each participant can efficiently retrieve node or resource information (data or services) located near any chosen geographic position. In such a system the responsibility for maintaining information about the position of active peers is distributed among nodes, in such a way that a change in the set of participants causes a minimal amount of disruption.

In this section we outline a framework for an overlay based on a Distributed Geographic Table, according to the core concepts and definitions of P2P systems presented in [14]

3.1 Conceptual Model and Neighborhood

In a generic DGT's overlay we define $\mathcal{P}$ as the group of peers, each one having an unique $id \in \mathcal{I}$ (where $\mathcal{I}$ is the space of identifiers) and a pair $(latitude, longitude)$. If we define $\mathcal{W}$ as the space of world's coordinates and $w \in \mathcal{W}$ as $w = (latitude, longitude)$ then a generic peer $p \in \mathcal{P}$ may be characterized by $\langle id_p, w_p \rangle$ where $id_p \in \mathcal{I}$ and $w_p \in \mathcal{W}$. The association between a peer in $\mathcal{P}$ and an identifier in $\mathcal{I}$ is established with the function $F_p : \mathcal{P} \rightarrow \mathcal{I}$.

In a DGT the distance between two nodes is evaluated as the real geographic distance between two world points (also known as great-circle distance or orthodromic distance) :

$$d : \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{R} \tag{1}$$

We define the *neighborhood* of a geographic point as the group of nodes that are geographically close to that specific position, that is they are located inside a given surrounding region. Defining $\mathcal{A}$ as the set of geographic regions delimited by a closed curve and $a_w \in \mathcal{A}$ a region centered in the geographic point w we can define neighborhood as:

$$\mathcal{N} : \mathcal{W} \times \mathcal{A} \rightarrow 2^{\mathcal{P}} \tag{2}$$

By $2^{\mathcal{P}}$ we mean the set of all possible connections between peers. In details, if we want to evaluate the neighborhood of a target geographic point $t \in \mathcal{W}$ using a region $a_t \in \mathcal{A}$ centered in t we can express $\mathcal{N}$ as:

$$\mathcal{N} = \{p \in \mathcal{P} | w_p \subseteq a_t\} \quad t \in \mathcal{W}, a \in \mathcal{A} \tag{3}$$

Where, as earlier, w_p is the geographic position of peer $p \in \mathcal{P}$. We can select for example a circular region $\mathcal{C} \in \mathcal{A}$ with a radius $c_r \in \mathbb{R}$ to evaluate the node's neighborhood. In this case c_t and $\mathcal{N}$ become :

$$c_t = \{w \in \mathcal{W} | d(w, t) \leq c_r\} \quad t \in \mathcal{W} \tag{4}$$

$$\mathcal{N} = \{p \in \mathcal{P} | w_p \subseteq c_t\} \quad t \in \mathcal{W}, a \in \mathcal{A} \tag{5}$$

3.2 Routing Strategy

The basic service of a DGT overlay is to route a request to find available peers in a specific area or more precisely to determine the neighboorhood $\mathcal{N}$ of a generic global position $w \in \mathcal{W}$.

Routing is a distributed process using the overlay network. We can model it as asynchronous message passing: $route(p, w, a)$ forwards a request about the geographic position $w \in \mathcal{W}$ to a generic peer $p \in \mathcal{P}$ with reference to a region $a \in \mathcal{A}$. A routing strategy can be described by a potentially non deterministic function:

$$\mathcal{R} : \mathcal{P} \times \mathcal{W} \times \mathcal{A} \rightarrow 2^{\mathcal{P}} \tag{6}$$

which selects at a given peer p its neighborhood $\mathcal{N}(w, a)$ for the target global position.

The routing process should be based on the evaluation of the region of interest centered in the target position. The idea is that each peer during a routing step selects nodes that it presumes to be located inside or close to the chosen area centered in the target point. If the contacted node cannot find a perfect match for the request, it should return the closest nodes from its routing table to the target position. This procedure can be used to maintain the local $\mathcal{N}$ of a single peer around its position or to schedule a query to find available nodes close to a generic target. The general aim of the approach is to have high knowledge of nodes that are close to the peer's position and step by step a reduced number of known contacts that will be used to forward specific geographic queries.

In a DGT-based overlay each peer owns and maintains a structured routing table based on distance. This table can be organized in different ways to support the chosen routing algorithm and to reduce the number of hops necessary to obtain the requested results. Properties of a routing algorithm in a DGT are characterized by their associated cost measures, such as the number of hops or exchanged messages, the probability of successful routing and the percentage of found peers compared to the actual number of nodes available in a specific area.

3.3 Maintenance Strategy

Participation of peers in an overlay network dynamically changes over time. In case of DGT, peers change very often their geographic position and can freely join or leave the network any time. These updates of position or churn can happen frequently for which the $\mathcal{N}$ of a generic peer $p \in \mathcal{P}$ may be highly dynamic. To maintain the structural integrity of the DGT each peer needs to schedule periodically a maintenance procedure that compensates topological changes of the network and position or generic failure connection related to churn. The practical usability of a DGT critically depends on the efficiency and the period of this kind of procedure minimizing the effort in terms of computational load and sent messages. In DGTs, consistency is not only related to disconnections or network communication, but it is also associated to the dynamic and constant change of position of available nodes and the consequent reconfiguration of neighborhood for each active peer in the network.

3.4 Security

A very important issue in real-time localization concerns security and privacy. In a DGT the only data that are shared among peers are their unique ids, their IP addresses and ports for the communication, as well as their GPS coordinates. At the DGT level no reference should exists to sensitive data (*e.g.* MAC address) that may allow for identifying a node's owner. The capability of finding peers that are active and close to a specific geographic position is obtained without adding any kind of personal data. Sensitive or potentially dangerous information may be added by applications that on the DGT that may store such data in their data structures, but this is a problem related to the application layer and as such it potentially affects any P2P scheme.

4. PROACTIVE NEIGHBOR LOCALIZATION

Within the DGT framework presented in section 3, we have designed a P2P protocol, inspired to the approach of Kademlia [4] for the localization of all nodes that are "geographically" (rather than "virtually") close.

Whenever a single active node in the system wants to contact other peers in its area (*e.g.* to provide or search for a service), it does not need to route additional and specific discovery messages to its neighbors (or to a supernode responsible for a specific zone) in order to find peers that are geographically close. Instead, it simply reads its neighbor list, that is proactively filled with "geographic neighbors".

Each node knows his global position (GP) retrieved with GPS system or with other localization technologies, and knows a set of real neighbors organized in a specific structure based on the distance that these nodes have in relation to the node's position.

The main goal of the protocol is to build and maintain an overlay where each node knows all the active nodes available in a geographic region in order to provide and realize specific applications and services. An example of application based on our implementation is a city monitoring system that uses decentralized nodes to monitor the traffic status of the city. By using this system there is no need to deploy powerful servers: light peers can be activated in strategic points convering the city area. Each of them can analyse its region of interest, monitor traffic conditions in real-time and evaluate peer positions in order to inform them about accidents and traffic jams, suggesting the best path for their users.

4.1 Data Structures

Each peer stores a set of lists of neighbors, called *GeoBuckets* (GB), each list being sorted according to distance to the GP of the peer itself. Such lists are regularly updated in order to have the latest node's position. This structure of buckets can be considered as a group of K different concentric circles, each having a different (application-specific) radius R_i and thickness r_i, with i integer $\in [1, K]$, for which Ri is the sum of previous r_i from 1 to i.

If there is a known node whose distance from the peer is larger than the radius of the last circle R_K, it is inserted in another list that contains the nodes outside the circle model.

Each node in the neighbor list is characterized by *GPS Position*:(the latitude and longitude retrieved with a GPS

system or with other solutions *e.g.* GSM cell-based localization), *IP Address* (allows to contact the node. It can be a public IP directly linked with the node or with a proxy), *UDP Port*(port used for the communications with that node) and *Number of known nodes* (used to compare two nodes that are at the same distance).

4.2 Network Join

When a new node comes into the network and wants to join the system it receives the first list of neighbors (from a *bootstrap node* or loading it from local cache) that can be used for the future researches and nodes discovery. The new peer sends a "Join Request" with his GP to the bootstrap that generate and send a new peer's list, based on the position provided by the new applicant, that contains the closest available known peers near these coordinates. The bootstrap sends up to L references to known peers (*i.e.* previously connected peers that are still alive). It is important to emphasize that these information are not updated, because referenced peers may have moved away from their initial location. This kind of operation is performed not only during the first join of a peer, but also if the peer finds itself to be almost or completely isolated. In this particular situations (frequent when peers enter low density areas) the node can send a new request to the bootstrap trying to find some new connected devices in his zone.

4.3 Node Discovery

The main procedure used during peer discovery is *findNodes(GP)*, that returns the β nearest nodes near a specific geographic position. This procedure is usually executed by a peer n upon a request from another peer. Node n searches in the geo-bucket associated to the requested GP. If that geo-bucket contains less than β entries, peer n searches all its geo-buckets.

4.4 Periodic Lookup

Each peer periodically performs a lookup procedure in order to find the $\alpha < K$ peers that are nearest to the selected GP. Such peer set may include newly connected nodes as well as mobile peers that have entered the visibility zone. The lookup initiator starts by picking α nodes from its closest non-empty GeoBucket (or, if that bucket has fewer than α entries, it just takes the α closest nodes). Such peer set is denoted as $C_i = \{n_{1i}, .., n_{\alpha i}\}$, where i is an integer index. The initiator sends parallel *findNodes* requests (using its GP as target) to the α nodes in C_i. Each questioned peer respond with β references. The initiator sorts the result list according to the distance of the target position. It picks α peers that it has not yet queried and re-sends the *findNodes* request (with the same target) to them. If a round of *findNodes*s fails to return a peer closer than the closest already seen, the initiator re-sends the *findNodes* to K closest nodes it has not already queried. The lookup terminates when the initiator obtained responses from the K closest nodes. A peer can run a new lookup procedure only if the previous one is completed in order to reduce the number of exchanged messages and avoid the overlapping of same kind of operation. These operations are summarized in algorithm 1.

The lookup ends after f cycles, each cycle resulting with an updated set of nearest neighbors C_i. Thus, the number of sent messages (*findNodes(GP)*) is $f \cdot \alpha + K$ that depends on the users distribution and on the density in the area of interest.

Algorithm 1 periodicLookup(GP)

```
1: i ← 0
2: get α nodes from geo-buckets (nearest to GP): C_i = {n_1i, .., n_αi}
3: repeat
4:    j ← 1
5:    while j ≤ α do
6:       if n_ji not yet queried then
7:          n_ji.findNodes(GP)
8:       end if
9:       j ← j + 1
10:   end while
11:   get α nodes (nearest to GP) from the αβ results: C_{i+1}
12:   i ← i + 1
13: until C_{i+1} == C_i
14: f ← i
15: get K nodes (nearest to GP) from geo-buckets, not already in C_f
16: j ← 1
17: while j ≤ K do
18:    if n_ji not yet queried then
19:       n_ji.findNodes(GP)
20:    end if
21:    j ← j + 1
22: end while
```

4.5 Position Update

Each single peer (A) active in the network can change its geographic position for many reasons (the user may be walking, driving, etc.). In order to improve the accuracy of peer's knowledge each node sends to its neighbors its GP's updates. In order to reduce the impact of this operation to node's computational operation and to the message's rate each single peer (A) performs the following two operations for each node (B) is its GBs before sending a position update:

- Peer A checks for the distance between itself and node B: $d_{AB} = dist(w_A, w_B)$. If such distance is bigger then R_K, it means that peer B is out of the visibility area of A and for this reason it removes B from its GB and sends to it a *RemoveMessage* in order to communicate the removing operation. This action is very important because if peer B does not receive this specific message, it keeps the reference of A in its GBs, but it does not receive new updates because A removed B from its GBs.

- Peer A checks for $d(A) = d(w_{A_{new}}, w_{A_{old}}) > \epsilon$. If the condition is true, n sends its position update to its neighbors. Parameter $d(A)$ allows to define the accuracy of update messages and can be configured according to application requirements. A low value of ϵ causes a high rate of exchanged messages, but a very high value reduce the accuracy of the peer's knowledge and damages the global performance of the protocol.

There is another important aspect, related to position updating, that we need to take into account. In order to improve the performance during the join procedure we decide to add an update message sent by each node to the bootstrap peers if the distance between its actual position and the one that it had when it entered the network is larger

then λ, *i.e.* $d_b(n) = dist(GP_{boot}(n), GP_{new}(n)) > \lambda$. These updates are performed only if the peer is moving far from its original area, and helps the bootstrap to provide more precise information to newcomers.

4.6 Gossiping

To improve the results and the performance of our protocol we added gossip information inside exchanged messages. This approach on one side increases the size of sent packets but at the same time adds significant knowledge that helps peers to be aware of available nodes in their area. In details, gossip information is attached to lookup messages (those that trigger the *findNodes* procedure). Each peer maintains the references to nodes that have been discovered during the time between two different lookup procedures. The propagation of this portion of knowledge among peers is naturally made by periodic lookup procedure performed each node trying to update their local neighborhood. This approach allows to reduce the overall number of exchanged messagges, at the cost of adding a little amount of data to the payload, because in this way we can obtain requested information in a reduced number of steps.

5. PERFORMANCE ANALYSIS

In this phase of the project, simulation is the most appropriated method to evaluate performance in terms of accuracy and scalability in different dynamic scenarios. We used the discrete event simulation tool called DEUS [6], that provides a simple Java API for the implementation of nodes, events and scheduling processes, and a straightforward but powerful visual tool for configuring simulations of complex systems.

In order the perform realistic tests we set up an integration between DEUS and Google Maps API. With the features provided by Google Maps API we have created a simple HTML/Javascript control page that allows to monitor any simulated node, following it from starting to final position, and all the neighbors in its GeoBuckets. This solution allows to study the protocol not only with specific P2P metrics - like message rate, miss ratio, number of peers, etc. - but also with a direct monitoring of peer behaviors during the simulation.

5.1 Peer Mobility Model

In our simulations each peer is a mobile node with a random base speed (ν_b) between $5Km/h$ and $100Km/h$ that can be associated for example to a pedestrian, or to car. In order to create some real scenarios we have generated offline a list of 3000 real paths, starting from 1000 random GPS coordinates centered in Frankfurt. Two random points were chosen from the list of available GPs, then with a specific Java application we generate a request to Google Maps web site to retrieve the path between them with all intermediate points. An active peer in the system selects a path and starts moving on it selecting step by step the new path's point. For each move, its speed is randomly selected according to an exponentially distributed random variable, with mean value ν_b.

5.2 Performance Metrics

We take into account the following metrics to evaluate the results of our simulations:

- *PMN*: Percentage of Missing Nodes in the GeoBuckets of a peer, with respect to those really present in the area.
- *MR [msg/sec]*: message rate, *i.e.* the number of received messages per seconds.
- *NPE [Km]*: Average of node position error, *i.e.* is the distance between a peer's position reference in a GeoBucket and the true position that the node really has.

5.3 Scenario 1

In the first scenario we want to study and evaluate relationships and dependencies between the number of GeoBuckets, the thickness of each list, the PMN and the MR.

Results are associated with two different kinds of simulation. The first one with a virtual time of 10h (10000 virtual time units) and 1000 available peers, and the second with 20h (20000 virtual time units) as simulation time, with 2000 nodes. In both cases the node set greatly increases until half simulation, after which a small but constant number of peers enters the network. The following table summarizes all the considered cases.

Case	#GeoBuckets	Thickness	#Peer	VT
1	10	1.5 Km	1000	10000
2	5	3 Km	1000	10000
3	10	0.5 Km	1000	10000
4	10	1.5 Km	2000	20000
5	5	1.5 Km	2000	20000

Fig.1 shows the PMN for cases 1,2 and 3, that refer to the first kind of simulation (10h and 1000 nodes). We notice that, on average, the PMN is very encouraging. In particular for the first half of simulation, where many new nodes enter the system, the value is around 5% and decreases when we are in a more stable situation like the second part of our test.

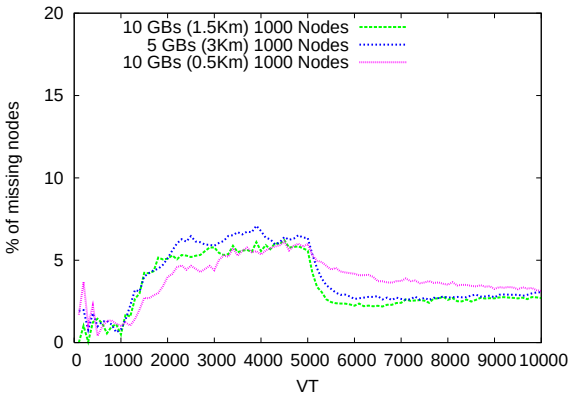

Figure 1: PMN - Scenario 1 (Cases 1,2 and 3) (with 1000 nodes).

Another important metric that we must take in account for these different configurations is the MR. Fig.2 shows results for cases 1,2 and 3. We know that a highly covered area ($\pi \cdot r_{GB}^2$) is linked with a potentially high number of active peers, *i.e.* an increased number of known nodes that we

must contact in particular for GP updates. For this reason, simulation results show that cases 1 and 2 have an increased MR value compared with case 3 where the covered area is smaller. In any case the number of exchanged messages is very low considering that it is decentralized system without the help of a central server, and knowledge is maintained by available peers.

We have performed the same kind of analysis with 2000 active peers and with two configuration of GeoBuckets (cases 4 and 5) in order to understand the protocol's behavior with a different distribution of nodes.

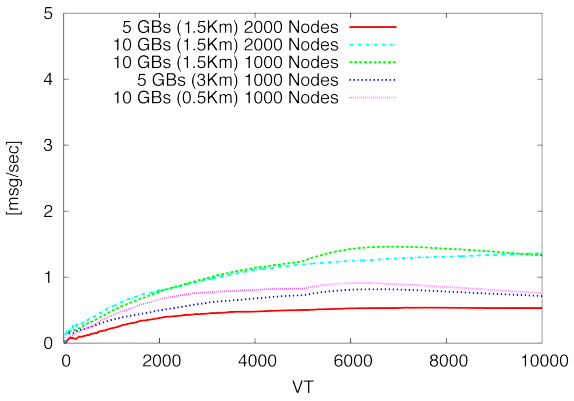

Figure 2: Global MR - Scenario 1

Results in Fig.3 show that also with a high number of available peers and with two sizes for the thickness of GeoBuckets, the PMN is very small (under 10% and around 5%). In case 4 we have 10 GeoBuckets with 1.5Km thickness, which means a covered area of 706Km^2 whereas in case 5 we have only 5 GeoBuckets with the same thickness for a covered area of 176Km^2. There is a great difference in the covered area but the performance is very good in both cases. This little amount of missing nodes depends on the dynamics created by new incoming peers (in particular during the first half of simulation where the percentage is lightly increased) and by the high rate of movements generated by available nodes that travel on their paths.

In order to provide this kind of performance with different configurations and covered area, the protocol needs to route messages to users in the target zone. These scenarios, as describes for previous results, imply a different amount of exchanged messages (Fig.2) that depends on the density of peers in the analyzed area.

We can say that the accuracy of the protocol shows little dependance on the configuration of GeoBuckets (number and thickness). Results show that also in different parameter setups we can have a very low PMN, considering also the high dynamic context of work where all peers are mobile users that change their position very often during the simulation. The other important aspect that comes from this analysis is the relationship between the covered area and the MR value, that we must take into account when we want to design an application based on that protocol in order to find the right compromise between the size of analyzed zone and the number of exchanged messages.

Another important issue related with the PNM is to understand the distribution of missing nodes in each available GBs in order to verify the knowledge's trend of active peers.

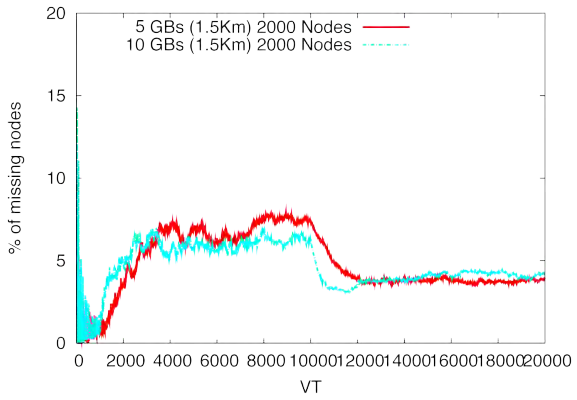

Figure 3: PMN - Scenario 1 (Cases 4 and 5) (with 2000 nodes).

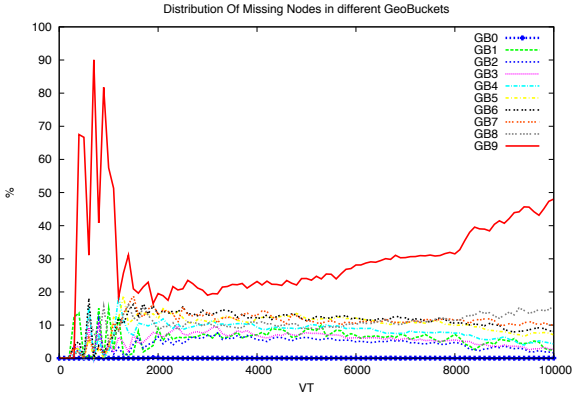

Figure 4: PNM Distribution in each GeoBucket - Scenario 1 (Case 1)

Fig.4 is related to case 1 with 10GB an a thickness of 1.5Km. We already showed the associated PNM in Fig.1 that globally results very low for all the simulation and around the 5%. Now we are analysing the distribution of this value in each different GB. We can see that for the first GeoBucket the percentage of missing node is around 0% for the whole simulation's time. As we can imagine the great amount of missing peers is located in the GB with an high index and that cover an area very far from the peer's coordinates. GB9 has the highest percentage of missing nodes and other GeoBuckets keep a value under the 20%. This is a very important result that shows how the protocol is very precise and reliable and how it respect the DGT idea to have an high percentage of known of peers that very close to a node's position. This result considering a thickness of 1.5 Km allow us to imagine other applications like for example Vehicular Networks where is very important to have the best knowledge of active users in a specific area of interest near the car.

5.4 Scenario 2

In this second scenario we evaluated the performance of our protocol in a context with an increased number of peers and with high dynamics created by the numerous joins. This simulation considers ≈ 5400 users, with a virtual time of 50h,

10 GeoBuckets with thickness of 1.5 Km. Fig.5 shows the percentage of missing peers that results a little bit increased if compared with the results of the first scenario, but in any case it is reasonably under the 10%.

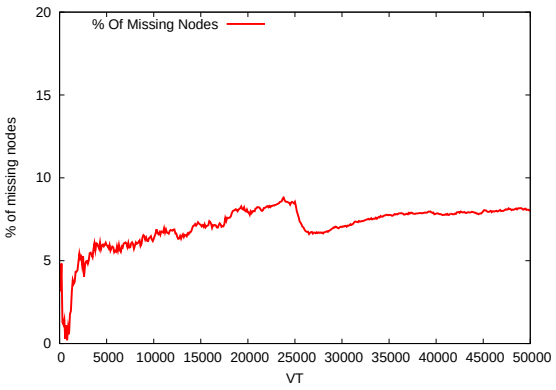

Figure 5: PMN - Scenario 2

The cost in terms of [msg/sec] (Fig.6) is very low if we consider that the covered area is very large and with an high density of active peers. We can see that in the first half of simulation there is an increase of the analyzed parameter because there are a lot of new joins in a small time and in the same area. This behavior creates new activities related to join and with peer position update that naturally generate new exchanged messages. In the second half, when the number of new incoming users is decreased the resulting MR is reduced.

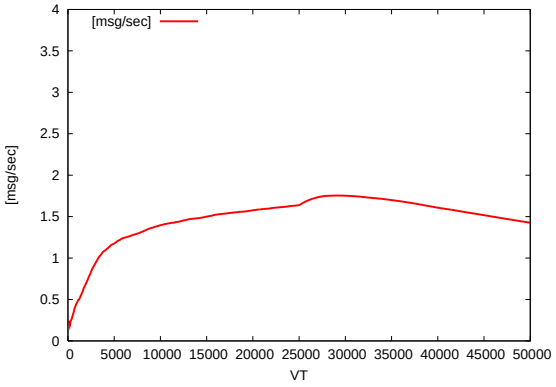

Figure 6: MR - Scenario 2

In this scenario for the first time we show the results associated with the NPE. The ϵ value that we use is of 0.5 Km, that is very low in particular if we consider the target area of each peer (10 GB * 1.5Km). The results shows that on average the error is around ϵ for the duration of the simulation. This parameter's configuration can be associated for example with an application that needs a high precision for a very large covered area like for example a system for monitoring roads or highways.

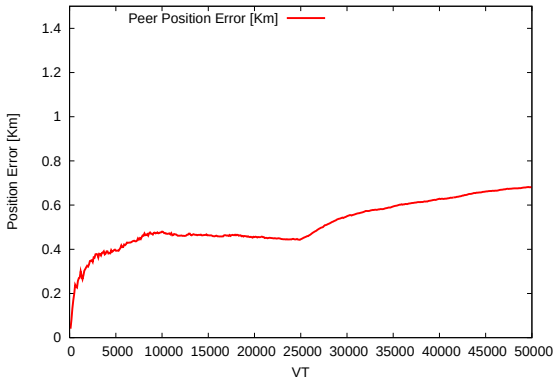

Figure 7: NPE - Scenario 2

5.5 Scenario 3

This last scenario was created to show the effect of ϵ variations on protocol performances. Using a network of 2000 node (10 GBs and thickness of 1.5Km), we realized six different kinds of simulation with a variation of ϵ between 0.1 Km to 1.35 Km with a step of 0.25 Km. As previously described ϵ is the threshold used to during the Position Update procedure. A low value means that the update of position is performed very often if the user change his/her location whereas an high value is linked with a rare update.

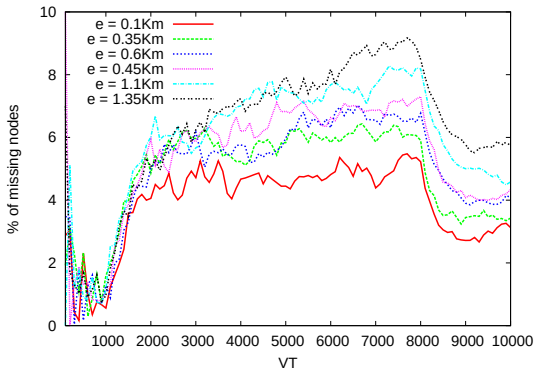

Figure 8: PMN - Scenario 3

Precision of information stored in GBs is clearly related to ϵ. Fig.8 shows the percentage of missing nodes with different kind of configuration and we can see that there is a noticeable variation in the results. This behavior is justified by the fact that a large ϵ value may lead to wrong exclusion or removal of a peer from the GBs, resulting into an accuracy loss and inconsistency.

The analysis of the NPE (Fig.9) shows that the average error is slightly larger than the threshold because there is a little variation introduced by peers' mobility and information's distribution among available nodes.

Another important aspect related to these analysis is the number of exchanged messages. A small value of ϵ that is results in a reduced error of position is strongly correlated with an increased value of MR. Fig.10 shows the results of different configurations and suggests that a value between 0.35 Km and 0.6Km can be a good compromise in terms of messages and accuracy for the chosen set of parameters.

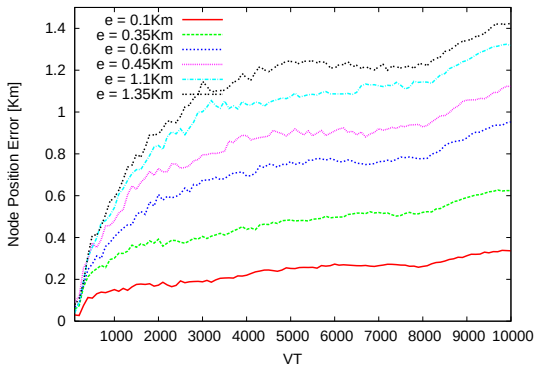

Figure 9: NPE - Scenario 3

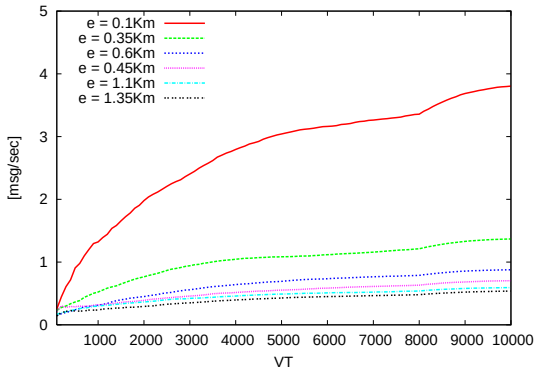

Figure 10: MR - Scenario 3

This last scenario allows to understand the importance of the ϵ parameter and how we can use it. Clearly this parameter is strongly related to the application, but with a good analysis there is the opportunity to reduce the number of exchanged message without a great impact on global precision.

6. CONCLUSIONS

Distributed localization for a massive number of users is a challenging problem, with many useful applications. In this paper we have introduced the concept of Distributed Geographic Table (DGT) referring to a decentralized system that allow to retrieve nodes or services located near any chosen geographic position using a distributed responsibility for maintaining information about the positions of active users. We presented also the particular instance of DGT, based on the peer-to-peer paradigm, that allow peers to localize all available nodes near to their geographic position enabling distributed and low cost application like city and traffic monitoring. We have shown that good performance can be achieved at low cost in terms of message rate. All parameters in our protocol can be tuned in order to achieve the most suitable performance for the considered application. The distribution of missing nodes in available GBs shows also that for the first container the percentage is almost zero and very low for the others GBs except for the last one. These results allow us to envision many different applications for the DGT, such as vehicular networks, where the region of interest may range from the area surrounding the car, to retrieve specific information about accidents, or to any remote point of interest, to find information like the traffic status.

We also plan to investigate a formal model to support the general analysis of our DGT implementation. Moreover, we plan to take into account the estimation of peer trajectory (*e.g.* nodes traveling along highways) in order to reduce the number of exchanged messages. Finally we intend to exploit local communication (Ad-Hoc networks) to directly retrieve available peers in the neighborhood and exchange useful information about GeoBuckets data.

7. REFERENCES

[1] A. Kovacevic, N. Liebau, R. Steinmetz, *Globase.KOM - A P2P Overlay for Fully Retrievable Location-Based Search* , Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing, September 2007.

[2] S. Asaduzzaman, G.v. Bochmann, *GeoP2P: an Adaptive and Fault-tolerant Peer-to-peer Overlay for Location Based Search*, CoRR 2009.

[3] A. Harwood and E.Tanin *Hashing Spatial Content over Peer-to-Peer Networks*, In ATNAC, page 5, 2003

[4] P. Maymounkov, D. Mazières *Kademlia: A Peer-to-peer Information System Based on the XOR Metric*, First International Workshop on Peer-to-peer Systems, 2002

[5] Google Latitude. `www.google.com/intl/en_us/latitude/intro.html`.

[6] M. Amoretti, M. Agosti, F. Zanichelli, *DEUS: a Discrete Event Universal Simulator*, in Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009

[7] M. Li, W. Lee, and A. Sivasubramaniam. DPTree: A Bal- anced Tree Based Indexing Framework for Peer-to-Peer Sys- tems. In 14th IEEE ICNP, pages 12-21, Nov. 2006.

[8] D. A. Tran and T. Nguyen. Hierarchical Multidimensional Search in Peer-to-peer Networks. Computer Communications, 31:346-357, 2008.

[9] B.Namand, A.Sussman. DiST:fullydecentralizedindexing for querying distributed multidimensional datasets. In 20th IPDPS, page 10, Apr. 2006.

[10] E. Tanin, A. Harwood, and H. Samet. Using a Distributed Quadtree Index in Peer-to-peer Networks. The VLDB Jour- nal, 16(2):165-178, 2007.

[11] B. Liu, W. C. Lee, and D. L. Lee. Supporting Complex Multi-Dimensional Queries in P2P Systems. Jun. 2005.

[12] A. Guttman. R-Trees: A Dynamic Index Structure for Spa- tial Searching. In SIGMOD84, pages 47-57, Jun. 1984.

[13] A. Balasubramanian, B. N. Levine and A. Venkataramani, *DTN Routing as a Resource Allocation Problem*, Proc. of SIGCOMMŠ07, Kyoto, Japan, August 2007.

[14] K. Aberer, L.O. Alima, A. Ghodsi, S. Girdzijauskas, S. Haridi, M. Hauswirth, *The essence of P2P: A reference architecture for overlay networks*, 2P2005, The 5th IEEE International Conference on Peer-to-Peer Computing, August 31-September 2, 2005, Konstanz, Germany.