Hovering Information: A Self-Organising Infrastructure-Free Information Storage and Retrieval Service

Alfredo A. Villalba Castro[1], Giovanna Di Marzo Serugendo[2] and Dimitri Konstantas[1]

[1]Department of Information Systems, University of Geneva
CUI-Bat A, 7 rt de Drize, 1227 Carouge, Switzerland
[2]Computer Science and Information Systems, Birkbeck (University of London)
Malet Street, London, WC1E 7HX, U.K.
{Alfredo.Villalba, Dimitri.Konstantas}@unige.ch
dimarzo@dcs.bbk.ac.uk

Abstract. This paper introduces the Spatial Memory concept which defines any geographical location as a memory where information can be stored and retrieval by user-applications. The Hovering Information Middleware is also introduced as a constrained implementation of the Spatial Memory concept. This middleware is based on results presented on previous works related with Hovering Information. Two main modules of the middleware are discussed and evaluated, the Storage and the Retrieval Modules. The evaluation is done by simulations using both random-way-point and real mobility patterns. Real mobility patterns are generated with the SLAW real mobility model. The simulations evaluate current storage and retrieval algorithms and outlines the major drawbacks and potential improvements for next milestones.

Keywords. Mobile services, Geolocalisation, Ad-hoc networking.

1 Introduction

Nowadays, we are witnessing the extraordinary progress of technology in terms of miniaturization and communication. Fixed and mainly mobile devices are much smaller and much more powerful than ten years ago. It has become normal for someone walking in the street to carry with him a mobile phone having as much *memory and computing power* as a personal computer of the nineties. Besides this fact, current devices (both fixed and mobile) possess *powerful wireless communication interfaces* which make them able to communicate with distant servers via the Internet or even to communicate between them in an ad hoc manner.

In the last decade, mobiles devices as they are becoming more powerful have started playing an important role enabling people to be connected not only between them (phone calls and text messages) but to the Internet and then to all the ad hoc services. In addition, the increasing development of geo-localized services leads us to the design of even more powerful services in mobile devices.

Besides the technological improvements and the undisputed success of the Internet, the scheme of how information is produced, stored and retrieved has been

Castro A., Di Marzo Serugendo G. and Konstantas D.
Hovering Information: A Self-Organising Infrastructure-Free Information Storage and Retrieval Service.
DOI: 10.5220/0004465500690087
In *Proceedings of the 4th International Workshop on Enterprise Systems and Technology (I-WEST 2010)*, pages 69-87
ISBN: 978-989-8425-44-7

also evolving from centralized schemas to *community and collective* ones. We are not any more limited to applications working with only one or a group of some powerful servers on which information (databases, videos, profiles,...) is stored. We are now facing services such as peer-to-peer sharing files (kaaza, bittorrent,...), user-produced streaming video (YouTube), and social networking (MySpace, Facebook, LinkedIn,...). These applications are different from classic ones in two points: either they are peer-to-peer and there is no more a centralized server, or the information they manage is produced in a collaborative and continuous way by the users.

However, mobile devices are still playing a passive role since most of them are consumers of information but not producers. It is becoming usual for us to send and receive emails, to browse and to navigate in a city thanks to the geo-localization in-built services, all this in our mobile phone. But still, these applications are only consumers of information and they have to get connected to a centralized server via the Internet. What is still missing is applications benefiting from the potential of collective work arising among mobile devices, sharing both information and computing. Sensor networks, for instance, follow this paradigm but we want to extend this vision to any set of mobile and stationary devices.

In addition, applications following the centralized scheme may face problems of network saturation or robustness against failures. Moreover, stigmergy-based, context-awareness or spontaneous social networking applications need to use information being meaningful only locally, for a limited period of time, thus making the use of (centralized) servers unnecessary. There is no need for the information to travel a large geographical region when both the producer and consumer of that information are next to each other.

In this paper, we introduce the Spatial Memory vision in which any region of the earth may be defined as a memory able to store and retrieve any kind of information. Spatial memory as a general concept may rely on any kind of technology from centralized servers to mobile devices, from infrastructure to mobile ad hoc networks. However, we propose a concrete implementation of spatial memory called Hovering Information which has been already presented in previous works 121314. In other words, we have extended the Hovering Information concept to the more general concept of Spatial Memory.

In addition to the conceptual definitions we present in this paper, we also present a middleware architecture for the Hovering Information implementation. We discuss about its requirements and properties. We also present the storage and retrieval algorithms which are the core of the middleware. Storage and retrieval have been evaluated through simulations using both random-way-point mobility pattern and real mobility patterns.

1.1 Related Work

The Virtual Infrastructure project 4567 defines virtual (fixed) nodes implemented on top of a MANET. This project proposes first the notion of an atomic memory, implemented on top of a MANET, using the notion of quorums or focal points where a reasonable amount of mobiles nodes intersect. The motivation behind this project is the development of a virtual infrastructure on top of which it will be easier to define or adapt distributed algorithms such as routing, leader election, atomic memory,

motion coordination, etc. Hovering information shares similar characteristics, it tries to benefit from the mobility of the underlying nodes, but the goal is different. We intend to provide a hovering information service on top of which applications using self-organising user-defined pieces of information can be built.

GeOpps 9 proposes a geographical opportunistic routing algorithm over VANETs (Vehicular Ad Hoc Networks). This work focuses on routing information to some geographical location, it does not consider the issue of keeping this information alive at the destination, while this is the main characteristics of hovering information.

The work proposed by 10 aims to disseminate traffic information in a network composed by infostations and cars. While the idea is quite similar to that of hovering information, keeping information alive in its relevant area, this study does not consider the problem of having a limited amount of memory to be shared by many pieces of information or the problem of fragmentation of information. It also takes the view of the cars as the main active entities, and not the opposite view, where it is the information that decides where to go.

The Ad-Loc project 1 proposes an annotation location-aware infrastructure-free system. Notes stick to an area of relevance which can grow depending on the location of interested nodes. Notes are kept in their relevance area is by periodically broadcasting location-aware information to neighbouring nodes. This work also proposes to use this annotation system as a cache for Internet files in order to spare bandwidth. In this case, URLs are used as note identifiers. Similarly to the previous work, nodes are the active entities. In addition, in this case the size of the area of relevance grows as necessary in order to accommodate the needs of users potentially far from the central location. The information then becomes eventually available everywhere.

The ColBack system 28 is part of the MoSAIC project and intends to set up a collaborative backup system for mobile devices. This system does not focus on geo-localized information but replication strategies and replica scheduling and dissemination techniques could be used as inspiration for hovering information replication algorithms.

PeopleNet 11 describes a mobile wireless virtual social network which mimics the way people seek information via social networking. It uses the infrastructure to propagate queries of a given type to users in specific geographical locations called bazaars. Within each bazaar the query is further propagated between neighbouring nodes via peer-to-peer connectivity until it finds a matching query. The proposed queries propagation inside bazaar techniques could be a source of inspiration when we will develop query to retrieve specific hovering information.

2 Spatial Memory

The main idea is that of making a geographical region to behave as a memory where information of any kind could be stored, modified and read by user applications. This information will be stored in devices (mobile and fixed) being inside the region. The environment is highly dynamic as devices can leave the region, fail, run out of memory or just be switched off; or new devices can enter into the region.

Let us develop a more global vision of the concept. We consider the surface of the earth as a whole geographical region in which plenty of computers, servers, mobile phones, PDAs, RFIDs, smartphones, iPhones, sensors, etc are storing and gathering information; processing it and producing still new information. The exchange of information between all of them is via the Internet or local networks using wireless or fixed technologies. Let us call all these devices as nodes and let us sub-divide the whole region in arbitrary sub-regions that we will also call them regions by simplicity.

Each region has potentially some storage capacity where information can reside; this capacity depends on the number, characteristics and state of devices being inside the region at some point in time. We can imagine that information could be of any type and size, from text to video and from some bytes to several megabytes (or even gigabytes).

Once a piece of information created, it will have a self-organizing nature. The information will be responsible of staying at the region with which it is associated, or to migrate from one region to another if requested. It will be responsible of being accessible by user applications that will be able to read and modify it. To achieve all this goals, information will use the storage, computing and communication resources of the nodes. Regions are passive elements of spatial memory and pieces of information are the active ones.

It is important to notice that one major difference between Spatial Memory and current information management systems (databases, servers, p2p) is the way of addressing information. Traditional systems associate an address to a server or a peer storing information through which we can reach the information on request. However, in the Spatial Memory concept the way of addressing information is by geographical regions. We ask for information having some type and content, but being stored at some particular region. However, it is also possible to ask for information having some characteristic without specifying its location, in this case it is up to Spatial Memory to figure out where the information is stored. In both cases, there is a strong relation between geographical location and information because the latest is anyway stored at some region.

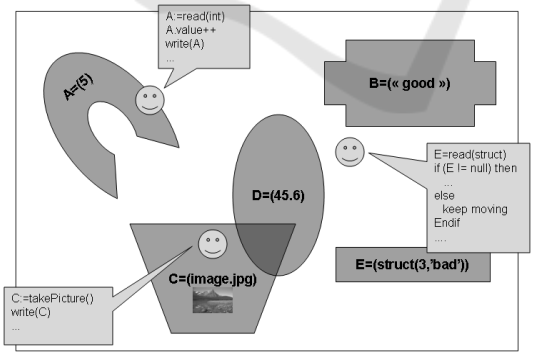

Fig. 1. Spatial Memory Vision.

User applications will only have to make use of simple primitives to basically read or write information from/to some region. Spontaneous social networking, disaster

areas, stigmergy-based and context-aware applications are the most appropriate for using this paradigm. Figure 1 depicts a geographical region where several pieces of information co-exist and each is associated with a specific region. We can also observe the presence of user applications that access the information of their interest in order to perform some tasks.

Extrapolating the vision of Spatial Memory to current information management systems (i.e. web, video/audio streaming, databases, etc.), we can consider this information as information attached to the regions of these servers. Whenever specific information is required by a client, the data will then migrate from the region of the server to the region of the client, where it is processed by user-applications. Another view is of considering the whole earth as the region of information stored in servers.

2.1 Requirements and Issues

Setting up a system such as Spatial Memory is undoubtedly not trivial since the environment is highly dynamic and unreliable. In the paragraphs below, we enumerate and discuss the most relevant requirements and issues.

Persistency. This is a major issue because of the dynamism and unreliability of the environment. In such conditions, an existing piece of information is likely to disappear when a node switch off or leave some region; a node runs out of memory; or some network errors happen. It is then extremely important that information find by itself the way to stay alive, to survive. In the other hand, time life of information depends on its nature. There might be some information being relevant only for some few minutes or hours; and another for some weeks or months. It is evident that short time life information fits better with this environment.

Reliability. While persistency deals with keeping information alive, reliability consists on keeping information accessible for nodes willing to read or modify it. Again, this is a very challenging issue to address. An existing piece of information may leave its meaningful region and be the last one piece on this area. Therefore, any node willing to access it will not be able to do it even though the information still exist. Although a fully reliability is the ideal situation, several applications might not need such a quality to still work well. For instance, instantaneous social networking applications may be more flexible in terms of reliability compared to disaster areas applications.

Consistency. Being Spatial Memory a highly distributed information storage system, inconsistency problems will follow. We can identify two sources of inconsistency. First, the fact that a piece of information will certainly exist in several nodes (distributed), a consistency issue arises when some node modifies its content and this modification must take some while to be known by other nodes hosting the same piece of information. Another situation is concurrent modification of a piece of information, two different version of the same information could exist at the same time. Last case, when the same information exist in two different and not overlapping regions, in this case each region could have a different version. In all these cases, ensuring consistency is a challenge. However, inconsistency in information could also be an advantage in some cases since human social information is most of the time contradictory (there is no an absolute knowledge) and society still works.

Atomicity. When writing, reading or modifying a piece of information, Spatial Memory must ensure the atomicity of these operations. Some information could be split in several pieces of information and spread in several nodes depending on the storage capacity of nodes. Later, this pieces will be merged into one piece of information to be used by applications. This kind of behaviour is definitely a challenge to overcome when dealing with atomicity. Another source of problems is communication failures but we could delegate this responsibility to the overlay network.

Trust. The nature of information is dynamic and user defined. These aspects are both an advantage and at the same time a source of trust challenges. We have to address challenges such as how to trust to some information published by some one else, or how to be sure that some information is not a trap for us. More an application relays on the reliability of the source of information, more it will sensible to the trust of information. As a first approach, we can image a kind of reputation systems in order to solve this problem.

Integrity. An application should be able to modify only its related information. In this way, integrity of information has to be conserved. It is in this context that attributes such as owner and access rights are aimed to exist. At the same time, it arises a problem of supplanting an identity and then modifying an information that normally we could not do it. Again, it is another challenging issue to solve when designing a concrete implementation of Spatial Memory.

Cooperation and Privacy. Spatial Memory requires a high degree of participation from nodes. But, this means that nodes have to be willing of sharing theirs storage, computing and networking resources. Otherwise, such a system will not work because it needs a critical mass of participants. Besides this problem of cooperation, it arises a problem of security and privacy. Participant nodes need enough guarantees that they will not be attacked because of being more open to cooperate, or that their private data would be accessible by anyone. A possible solution could be setting up a virtual machine on nodes to create a fully protected space where spatial memory could work.

3 Hovering Information Middleware

This section outlines a particular design of the Spatial Memory concept described above. Leveraging on our previous work on hovering information 1314, we show here how to build such a service through a middleware called Hovering Information Middleware (HIM).

3.1 Global Overview

The middleware assumes the presence of mobile (fixed) nodes moving around a geographical 2D area. Each node is able to store and process information, that we call it hovering information, as well as sent it to and receive it from neighbouring nodes in an ad hoc manner. User-applications are able to create information to be stored at a specific geographical region chosen by the application itself or by the middleware.

The creation process of information could be both triggered by a human user or by the application itself in an automatic way.

Figure 2 depicts several applications running on different groups of nodes; each application interacts with the hovering information middleware in order to perform its task.

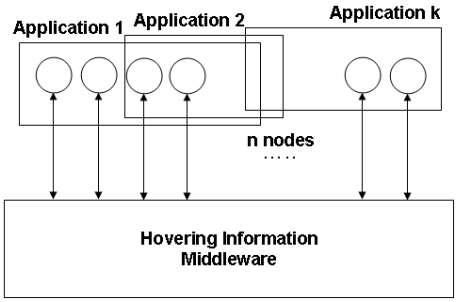

Fig. 2. Applications Deployment.

A piece of hovering information can be of any type from simple raw text to videos. Once created, a piece of information might be attached to a precise geographical region defined by the user-application. If not, it is up to the middleware to find a region where to store it. In both cases, information can be retrieved from anywhere unless strict region-bounded constraint is specified.

From an application perspective, hovering information is a concept characterising self-organising information responsible to find its own storage on top of a highly dynamic set of mobile devices. A piece of hovering information is attached to a specified location, which we call the anchor location. The main requirement of a piece of hovering information is to keep itself stored in the vicinity of the anchor location, which we call the anchor area, despite the unreliability of the device on which it is stored. Whenever the mobile device, on which the hovering information is currently stored, leaves the area around the specified storage location, the information has to hop - hover - to another device.

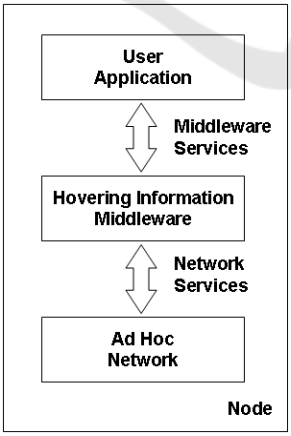

Fig.3. Node Architecture.

Pieces of hovering information have to be seen as information particles "suspended" into the given geographical region, being *attracted* by their anchor location. A piece of hovering information comes with a unique identifier, the actual data value to store and additional policies for spreading, self-destruction, etc. By means of simplicity, we will use the term information or hovering information in a exchangeable way.

From a conceptual point of view, the middleware itself is responsible of two main tasks: storing information (persistency) and retrieving it. We assume that the middleware use network primitives such as sending and receiving data, and services such as discovering neighbouring devices. On top of this, applications make use of the middleware services. Figure 3 illustrates these three layers in a node and their relation.

The current middleware we propose is designed to work in mobile ad hoc networks. In previous works 1314, we studied the behaviour of some simple but fundamental algorithms being responsible of the basic mechanism of keeping information persistent in a highly dynamic environment. Sub-section 4 discuss more in detail these algorithms.

3.2 User-application Primitives

The way how an application interacts with the middleware is through an Application Protocol Interface (API). Besides basic primitives for writing and reading information, the middleware proposes primitives for subscribing an application to some information-related events; and for defining regions, filters and policies. Let us explore more in detail each of these primitives.

1. *write(h, a, p)* – This primitive allows an application to write a piece of information *h* into a region *a* having the *p* policies.
2. *write(h, p)* – This primitive allows an application to write a piece of information *h* having the *p* policies. It is to the middleware to define the most appropriate region where *h* will be written.
3. *read(q, a)::h* – This primitive allows an application to retrieve a piece of information *h* matching the query *q* and being stored into the region *a*. If no information matches the query, an null value is returned.
4. *read(q)::h* – This primitive allows an application to retrieve a piece of information *h* matching the query *q*. It is up to the middleware to decide where to look up for this information. If no information matches the query, an null value is returned.
5. *subscribe(h)* – This primitive allows an application to automatically be notified when information *h* has been modified or removed.
6. *notify()* – This primitive is called by the middleware to inform to an application that a subscribed information has been modified or removed.
7. *region(w)::a* – This primitive allows an application to define a geographical region by its vertices *w*. A priori, the geometrical form of this region should be any. A region will be the semantic scope of a piece of information. The definition could be also done by passing the center and the radius of the region, in which case the region will a circle.
8. *applyRegionOperator(operator, a, b)::c* – This primitive allows an application to

combine two regions *a* and *b* and produce a new one *c* which will be the results of an *operator* such as union, intersection or subtraction among others. This kind of operators may be useful when extending the semantic scope of some information for instance.

9. *defineFilter(expression)::f* – Existing several pieces of information stored in the global region and many of them being stored in the same region or in overlapped ones, it should be possible to define a filter *f* when reading some information. This filter would define by an *expression* the region or regions, the type of information, the user id application, the owner id, the content, the author id, etc.

3.3 Middleware Architecture

The hovering information service is designed as a middleware service. It is a distributed service and each node must implement it. The Figure 4 outlines the proposed architecture.

The core of the proposed middleware are the Storage and Retrieval modules, both coordinate their behaviour according to the network properties (i.e. network saturation), local device properties (i.e. node routines), environment properties (i.e. neighbours state) and application requirement (i.e. low delay response when reading an application related information).

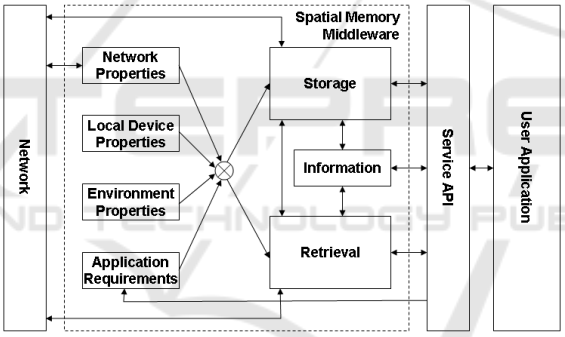

Fig. 4. Hovering Information Middleware Architecture.

Storage. This module is responsible of storing information in a region. It has as inputs data such as the network state, the local node state and the neighbours state. This data is provided by the respective modules. Some previous research has been done for this module and two algorithms have been proposed: attractor point and location-based caching 14.

Retrieval. This module is responsible for looking for an information requested by an user application. The information is searched in a particular region and filtered after the defined filter. The simple case is when information is stored at the current region, otherwise a process of propagation of the query will be triggered and the requested information, if found, will migrate to the querying area of the user application.

Network Properties. This module provides a transparent access to the network properties and current state such as traffic congestion, bandwidth, delay and current

neighbours among others. These network characteristics are used by the Storage and Retrieval modules in order to adapt their behaviour.

Local Device Properties. Local information of the device such as memory size, current position, current speed and prediction of location is made available by this module. This information is part of the set of parameters that Storage and Retrieval modules take in consideration to adapt their behaviour.

Environment Properties. This module makes available some other environment parameters such as neighbours nodes and their characteristics such us their current position and velocity vector.

Application Requirements. This module provides to the Storage and Retrieval modules parameters defining the expected performances an user applications requires to work in the right manner. Modules should have to adapt their efforts in order to reach these requirements as best as possible.

Information. This module is responsible of managing and processing the hovering information that is stored or retrieved from a region. It deals with attributes such as the age of the information, generation, priority, type, owner, etc. Meta-data such as replication time, version, creation time and generation are also processed in this module in cooperation with Storage and Retrieval modules (cf. §4).

3.4 Middleware Properties

The proposed middleware's properties come out from the design of the Storage and Retrieval modules and from the nature of the context which is highly dynamic and unreliable.

Infrastructure-Free. There is no need of infrastructure for this service. It is a kind of opportunistic service that profits from the presence of devices being around to do its job. It does not rely on any centralized server.

Self-Organisation. Once created and introduced in a region a piece of information, this last is completely autonomous. It looks by itself to survive and be accessible for user applications. The kind of operations that information could be able to do for this goal include replication, propagation, splitting, merging, swarm behaviour, etc.

Best-Effort Service. Being the environment highly dynamic, the service provided by spatial memory is not fully reliable and user applications should be designed having in mind this characteristic. However, user applications are able to specify their minimal requirements in order to achieve their goals.

Migration. We emphasize the fact that semantic scope of information is bounded to some region. However, it might be possible to replicate this scope some where else or to be expanded. In this case, semantic scope should be extended and modified and then a migration of information should be done.

Garbage Collector. Pieces of hovering information come with policies. Those policies prescribed a time-to-live limit after which the information self-destroys. This starts counting from the last time the information has been accessed or modified. This however means that the data is in fact modified to remember the last time of access.

Load-Balancing. The place where a piece of hovering information is stored is decided by either the application or the middleware (if not specified an specific region). In both cases, information replicates itself in quest of survive and competes with other information. Therefore, several different pieces of information may co-exist and previous results 14 have shown that geographical region is fairly distributed among different pieces of information.

4 Storage and Retrieval Algorithms

In this section we describe the current design of the storage and retrieval modules. We discuss about the main issues when designing them, the current implementation and its drawbacks, as well as the next milestone improvements in order to design a fully working middleware.

4.1 Storage Module

The information storage module is responsible of storing pieces of information, created by user-applications, in the most appropriate geographical region. As we mentioned in §2.1, keeping information alive and available are major challenges of such a system because of the highly dynamism of the environment.

Current Storage Algorithms. In previous work we have already presented two storage algorithms, called as replication algorithms: attractor point (AP) and broadcast-based (BB). We present the main concepts related to these algorithms.

Once a user-application creates a piece of information, the hovering information middleware will store it in the most appropriate geographical area called the *anchor area*. This piece of information can be tightly related to a geographical area (e.g. a context-aware information) or not, in which case the middleware can store information anywhere. In this paper, an *anchor area* is defined as a circle having a center, the *anchor location*, and a radius, the *anchor radius*, both defined by the user application to an specific piece of information. For simplicity we assume in this paper that an *anchor area* is always defined as a circle, but we could imagine that it could have any shape than a circle.

The AP and BB algorithms are based on self-replication. When a replica is in risk of disappearance it starts replication itself. To this aim, an *anchor area* and its surroundings are divided in four continuous areas, all of them having as center the *anchor location*. The *secure area* which is a disc having a radius smaller than the *anchor radius*, therefore this area is inside the *anchor area*. The *risk area,* which is the prolongation of the *safe area* beyond the *anchor area*, is a ring having a minimal radius equal to the radius of the *safe area* and a maximal radius greater than the *anchor radius*. The *meaningful area* is also a ring having a minimal radius equal to the maximal radius of the *risk area* and a maximal radius being normally quite far away from the *anchor location*. And finally the *irrelevant area* is the rest of the rest of the area where information is likely not relevant.

When a replica is in the *risk area*, it starts replication itself constantly in the quest of surviving. While the BB algorithms replicates a replica into all the neighbours of the respective host, the AP algorithms tries do it better by targeting only those neighbours being the k-closest ones (replication factor) to anchor location. We define the following attributes: replication factor (k_R), replication timer (T_R) and cleaning timer (T_C). A replication being in risk replication itself each T_R seconds and a node removes irrelevant replicas (those being in the *irrelevant area*) each T_C seconds.

Besides the two replication algorithms, we have also introduced two caching policies: the location-based caching (LBC) and the generation-based caching (GBC). In fact, in a context where several applications co-exist and each of them has several pieces of information (several context variables), it might happen that replicas compete for storage resources. That is why caching policies have been defined in order to choose the most appropriate replicas when storage space is not enough in a hosting node. The LBC policy is the most interesting and the most appropriated after our previous results 14. It is based on the location of the competing replicas and the hosting node. The most relevant replica to the host's location will have the priority.

Drawbacks of the Current Storage Algorithms. Previous replication algorithms and caching policies are not good enough yet. Although, they achieve their goals and their performances are good, the mayor drawback resides on the network overhead. The following paragraphs discuss these issues.

Once a replica is in its risk area, the replication frequency depends on the network saturation, the current state of host node's neighbours, the life cycle of the replica, ans the kind and priority of the information. This means that the replication mechanism adapts its behaviour to the saturation of the network in order to reduce the bandwidth consumption. Concerning the state of neighbours, a host node is constantly receiving messages from its neighbours, some of them are related to hovering information. In this way, by receiving some replication messages from away, a host node can estimate which replicas are already in its direct neighborhood, and a replication reduces its replication if there are several copies of it around. Concerning the lifecyle, a replica would have the tendency to replicate itself very often at the beginning of its existence, and then will gradually reduce this frequency assuming that the are enough replicas out there. Finally, a higher priority information will invest more effort on replication itself.

The current design of the replication algorithms is thought limited. A replica replicates itself when being in its risk area and keeps doing while being there with a well defined replication frequency. Therefore, we can straight notice the main drawback: network overhead. The current host of a replica may slowly pass through the risk area, or event stay there, and the replica would keep replicating itself permanently. An stop mechanism is needed. Moreover, the replication frequency is constant but we could imagine that this may evolve along the time as well. A compulsive replication might be suitable at the beginning of a prior of risk but not after a while of estimating that survival is quite guaranteed.

The current AP replication algorithm replicates a replica potentially in risk to a set of its neighborhood nodes being the closest ones to the anchor area of the replica. However, this implies that a multicast or several unicast messages are sent. The question that arises from this implementation is that broadcasting the replica to all neighbours might be more efficient and more or less equally expensive in terms of

power.

Previous results shows that the current replication algorithms get good performances in terms of keeping hoverinfos alive an available. However, the major drawback is the network overhead even though the AP does much better than a flooding scheme. From all this, we conclude that, and as stated and thought before, a replication decision based on the nodes behaviours may significantly improve and reduce the network overhead. The main idea reside in classifying nodes in two categories. Those who are most suitable to stay for a long time in a geographical area and those that not. The former would be the most suitable for storing replicas related to the respective location.

4.2 Retrieval Module

Once a mobile application or another system has created a hovering information and stored it in a geographical area, we must be able to retrieve this hoverinfo when required. We distinguish two kinds of retrieval mechanisms: a passive one and an active one. While in the *active retrieval* the user application asks for some information, in the passive *retrieval* the informations looks for matching user applications (publish/subscribe scheme). The answer message, if the query matches, propagates following a unicast scheme. In this paper, we only cope with the *active retrieval*.

We describe the active query propagation. Once a user application has asked for some information, the respective node generates a query that will propagate in quest of matching some replica stored in some node's buffer. The approach we present here deploys is a mitigated and geographically bounded flooding when the query is in the anchor area of the respective information. Otherwise, the query is propagated following a geographical routing until it gets to the anchor area of the information and then the previous method is triggered.

Each time a query arrives to a node, the nodes verifies whether the query has been already received in which case the node drops the query. Otherwise, the node verifies if the query matches any of the hoverinfos stored in its buffer to later send this hoveringo back to the demanding node. If the query does not match any hoverinfo, it propagates itself towards neighboring nodes. The decision of propagation is taken following an uniform distribution and a threshold previously defined.

Once a query has found a node containing a replica matching the query, the node sends back to the node source of the query an answer message with the respective hoverinfo. To route the message back, the node sends an unicast message to the node from where it received the query previously. Each node receiving this message behaves in the same way until the message gets to the destination node. Another way of sending back an answer without using the historic information is by only performing a geographical routing.

5 Evaluation

We evaluated the behavior of the above the current storage and retrieval modules

under different scenarios by varying the number of nodes, the number of hovering informations and the number of queries.

We performed simulations using the OMNet++ network simulator (distribution 3.3) and its Mobility Framework 2.0p2 (mobility module) to simulate nodes having a simplified WiFi-enabled communication interfaces (not dealing with channel interferences) with a communication two different communication ranges of 30m and 120m. Concerning the mobility module, an update interval of 1s and a window time of 10s has been used.

5.1 Simulation Settings and Scenarios

The generic scenario consists of a surface of 500m x 500m with mobile nodes moving around following a Random Way Point mobility model with a speed varying from 1m/s to 10m/s without pause time, or following a Real Mobility Pattern after the SLAW model 15.

In the generic scenario, pieces of hovering information have an anchor radius (r) of 50m, a safe radius (r_{safe}) of 30m, a risk radius (r_{risk}) of 70m, a relevance (meaningful) radius (r_{rel}) of 200m, and a replication factor of 4 (k_R).

Each node triggers the replication algorithm every 10 seconds (T_R) and the cleaning algorithm every 60 seconds (T_C). Each node has a buffer capacity of up to 20 different replicas. The caching algorithm is constantly listening for the arrival of new replicas.

Based on this generic scenario, we defined specific scenarios with varying number of nodes: from 20 to 200 nodes, increasing the number of nodes by 20; varying number of different pieces of hovering information existing in the system: from 20 to 200 hoverinfos, increasing the number of pieces by 20; and by producing 5 queries per hoverinfo created. Each of this scenarios has been investigated with different replication algorithms and the LBC caching policy.

We have performed 10 runs for each of the above scenarios. One run lasts 3'600 simulated seconds. All the results presented here are the average of the 10 runs for each scenario, and the errors bars represent a 95% confidence interval. All the simulations ran on a Linux cluster of 32 computation nodes (Sun V60x dual Intel Xeon 2.8GHz, 2Gb RAM).

5.2 Critical Mass

Figure 5 outlines the average availability of the AP and BB replication algorithms combined with the LBC caching policy. An average availability value has been computed for several scenarios containing each a different number of nodes.

We can observe that as expected the BB algorithm has in all the cases an average availability bigger than that of the AP algorithm. However, for this latest, we note an strange behaviour in comparison to previous results 1314, the availability does not keep growing as the number of nodes increases but it starts decreasing after 80 nodes. There are two issues for these results: either they are incorrect and the whole simulations should be verified, or any some missing detail is missing and they are correct. In the latest case, the critical mass which is defined as the minimal number of

nodes per area unit so that some acceptable level of availability can be reached (this threshold depends on the user-application but for the moment we assume that an acceptable one should bigger than 90%) would be around 100 nodes for the BB algorithm, and there would not exist for the AP algorithm since average availability is lower than 80%.

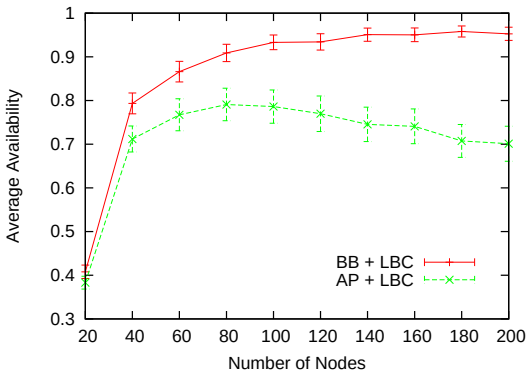

Fig. 5. Availability.

5.3 Network Overhead

Figure 6 depicts the number of sent messages for both the AP and the BB replications algorithms under several scenarios, each having a different number of nodes.

Again, in comparison to the results of the critical mass, we observe an strange behaviour for the AP algorithm. The number of sent messages is bigger than that of the BB for the range 20 to 140 nodes, and it starts decreasing after 80 nodes. These results are also incompatible with those of previous works 1314.

We define the network overhead as the number of messages exchanged between nodes per unit of time. One goal of storage and retrieval algorithms is to minimize this variable while maximizing the storage/retrieval performances. However, since our results seem to be biased, we can not conclude anything for the moment than verifying current results.

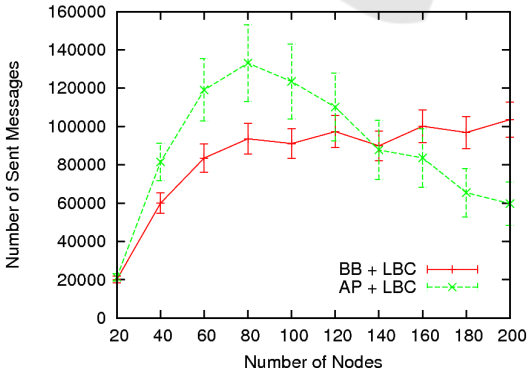

Fig. 6. Sent Messages.

5.4 Absorption Limits

Figure 7 shows the average availability for both the AP and the BB replications algorithms under several scenarios each having a different number of hovering informations but always either 100 or 200 nodes.

A hovering information system has limits. We can not store information as much as we want because of some inherent limitations: the whole storage capacity which is defined as a function of the number of nodes and the buffer size at each node. However, this absorption limit is below of this theoretical limit because of the redundancy of replicas.

The results of this paragraph shows that and absorption limit is reached for the BB because we can observe that the number of hoverinfos created in the system is not bigger than 120 hoverinfos. We can explain this results by the fact that nodes' buffer have a limit of 20 hoverinfos and that at some point they get full and no more hoverinfos are accepted. In the other hand, for the AP algorithm, we observe that a bigger amount of hoverinfos is absorbed by the system. However, these results should be verified because of the strange behaviour of the AP algorithms noticed in previous sub-sections.

5.5 Communication Range

Figure 8 depicts the average availability for both the AP and the BB replications algorithms under several scenarios having as main difference the communication range: 30m and 120m.

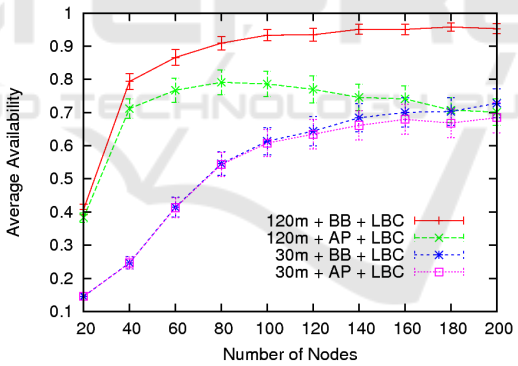

Fig. 7. Availabity and Communication Range.

We observe that as expected the availability is lower for a communication range of 30m. However, and in opposition to previous sub-sections, we notice that the behaviour of the AP in a configuration of 30m is more reliable and consistent with what has been expected and what has been found in previous works 1314. This fact leads us to think about the correctness of the results for 120m of communication range.

5.6 Real Mobility Patterns Scenarios

The previous results have studied the behaviour of hovering information middleware under an hypothetical scenarios on which every node moves randomly. This is quite rare to happen in reality, unless we consider a market or festival where we could observe a more chaotic behaviour in the movement of people. Therefore a set of simulations has been performed using real mobility traces or more precisely pseudo real mobility patterns, which have been generated by the SLAW mobility model 15.

Several scenarios have been defined each having different density of people moving around. Figure 9 presents a mobility trace of 200 nodes moving after the SLAW model, and an histogram of the locations by where nodes pass through. We can clearly observe, from both the trace and the histogram, that there are between 4 and 6 hotspots which could be considered as points of concentration of people. For instance in a real situation, these focal points might be considered as the main building of a university campus.

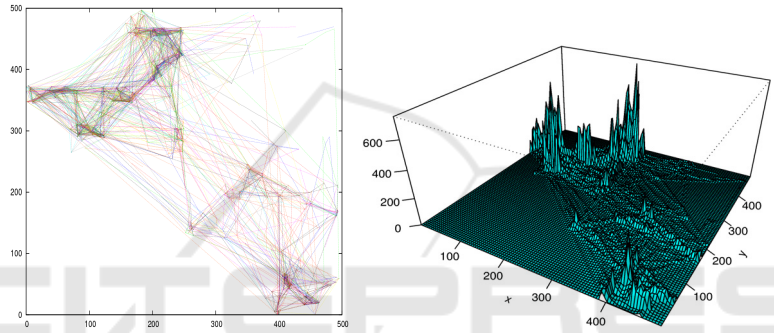

Fig. 8. Trace and histogram for SLAW real mobility model.

Figure 10 depicts the trace and histogram of 200 nodes moving after a random-way-point mobility pattern which has been extensively used during this technical report and previous works. We can notice that nodes have the tendency to pass more time in the middle of the area rather than in the sides. We can conclude from this that hoverinfos being in the middle will have more chances of surviving than those being created in the sides of the area. In the case of the real mobility patterns, the similar reasoning could be applied to the focal points.

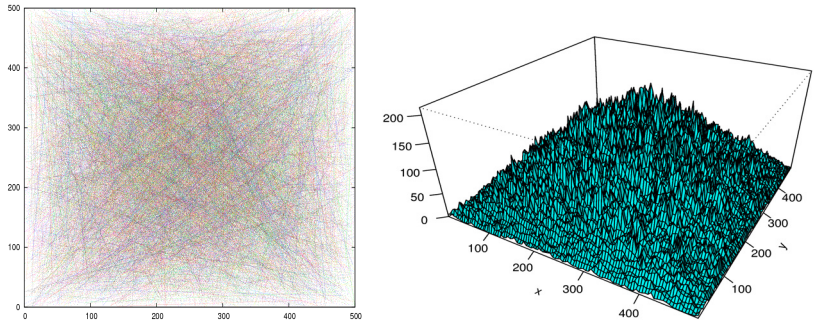

Fig. 9. Trace and histogram for random-way-point mobility model.

Figure 11 outlines the average availability for the AP and the BB replication algorithms under several scenarios having different number of nodes, which move after the SLAW real mobility model. We observe that for more than 80 nodes, the average availability for both algorithms is bigger than 60%, and for the BB algorithms is bigger than 80% which is not probably an acceptable value but it is not bad. For less than 80 nodes results are bad or strange, again further verifications and improvements have to be done.

6 Conclusions

In this technical report, we have described the Spatial Memory vision which defines any geographical region as a memory being able to store and retrieve any kind of information. We have also presented the Hovering Information concept re-engineered as a middleware implementing a constrained version of the spatial memory concept.

We have defined the API and the architecture of the Hovering Information Middleware, and described its properties. We have notably discussed about the current state and the major drawbacks of the Storage Module's replications algorithms, the Attractor Point (AP) and the Broadcast-based (BB), which were introduced in previous works. We have also introduced the Retrieval Module which for the moment implements a basic version of queries propagation, matching, and responses propagation back in a very simple way.

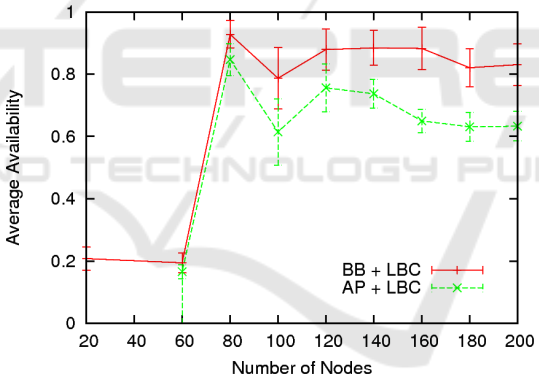

Fig. 10. Real Mobility Patterns – Availability.

We have run a set of several simulations using random-way-point mobility model and SLAW real mobility model. However, some results are strange and further verification is required, maily for the AP algorithm. In what concerns real mobility patterns, results are encouraging because of the quite high performances for the BB replication algorithm.

It is clear that some improvements are needed in the storage and retrieval algorithms. One of them, and most probably the most interesting one, goes by adapting the behaviour of the algorithms to the daily behaviour of nodes, for instance to the nodes' mobility routines.

References

1. D. J. Corbet and D. Cutting. Ad loc: Location-based infrastructure-free annotation. In ICMU 2006, London, England, Oct. 2006.
2. L. Courts, M.-O. Killijian, D. Powell, and M. Roy. Sauvegarde cooprative entre pairs pour dispositifs mobiles. In UbiMob '05: Proceedings of the 2nd French-speaking conference on Mobility and uibquity computing, pages 97–104, New York, NY, USA, 2005. ACM Press.
3. G. Di Marzo Serugendo, A. Villalba, and D. Konstantas. Dependable requirements for hovering information. In Supplemental Volume - The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pages 36–39, 2007.
4. S. Dolev, S. Gilbert, L. Lahiani, N. A. Lynch, and T. Nolte. Timed virtual stationary automata for mobile networks. In OPODIS, pages 130–145, 2005.
5. S. Dolev, S. Gilbert, N. A. Lynch, E. Schiller, A. A. Shvart sman, and J. L. Welch. Virtual mobile nodes for mobile ad hoc networks. In DISC, 2004.
6. S. Dolev, S. Gilbert, N. A. Lynch, A. A. Shvartsman, and J. Welch. Geoquorums: Implementing atomic memory in mobile ad hoc networks. In DISC, 2003. .
7. S. Dolev, S. Gilbert, E. Schiller, A. A. Shvartsman, and J. Welch. Autonomous virtual mobile nodes. In DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing, pages 62–69, New York, NY, USA, 2005. ACM Press.
8. M.-O. Killijian, D. Powell, M. Banâtre, P. Couderc, and Y. Roudier. Collaborative backup for dependable mobile applications. In MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, pages 146–149, New York, NY, USA, 2004. ACM Press.
9. I. Leontiadis and C. Mascolo. Geopps: Opportunistic geographical routing for vehicular networks. In Proceedings of the IEEE Workshop on Autonomic and Opportunistic Communications. (Colocated with WOWMOM07), Helsinki, Finland, June 2007. IEEE Press.
10. I. Leontiadis and C. Mascolo. Opportunistic spatio-temporal dissemination system for vehicular networks. In MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking, pages 39–46, New York, NY, USA, 2007. ACM Press.
11. M. Motani, V. Srinivasan, and P. S. Nuggehalli. Peoplenet: engineering a wireless virtual social network. In MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking, pages 243–257, New York, NY, USA, 2005. ACM Press.
12. A. Villalba and D. Konstantas. Towards hovering information. In Proceedings of the First European Conference on Smart Sensing and Context (EuroSSC 2006), pages 161–166, 2006.
13. A. Villalba Castro, G. Di Marzo Serugendo, and D. Konstantas. Hovering information - self-organising information that finds its own storage. In IEEE International Conference on Sensors, Ubiquitous and Trust Computing (SUTC'08), 2008
14. A. Villalba Castro, G. Di Marzo Serugendo, D. Konstantas, Hovering Information - Infrastructure-Free Self-Organising Location-Aware Information Dissemination Service, 2nd ERCIM Workshop on eMobility in conjuntion with WWIC 2008, May 2008.
15. Kyunghan Lee, Seongik Hong, Aeong J. Kim, Injong Rhee, Song Chong, SLAW: A Mobility Model for Human Walks, In Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), April 2009.