Infrastructureless Spatial Storage Algorithms

JOSE LUIS FERNANDEZ-MARQUEZ, Artificial Intelligence Research Institute, Spanish National Research Council
GIOVANNA DI MARZO SERUGENDO, Birkbeck College, University of London
JOSEP LLUIS ARCOS, Artificial Intelligence Research Institute, Spanish National Research Council

This article defines and analyzes a collection of algorithms for persistent storage of data at specific geographical zones exploiting the memory of mobile devices located in these areas. Contrarily to other approaches for data dissemination, our approach uses a viral programming model. Data performs an active role in the storage process. It acts as a virus or a mobile agent, finding its own storage and relocating when necessary. We consider geographical areas of any shape and size. Simulation results show that our algorithms are scalable and converge quickly, even though none of them outperform the others for all performance metrics considered.

Categories and Subject Descriptors: C.2.4 [**Computer Communication Networks**]: Distributed Systems; C.2.3 [**Computer Communication Networks**]: Network Operations; B.3.0 [**Memory Structures**]: General

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Microcomputers, portable devices, mobile code, spatial computing, data dissemination, wireless networks

ACM Reference Format:
Fernandez-Marquez, J. L., Di Marzo Serugendo, G., and Arcos, J. L. 2011. Infrastructureless spatial storage algorithms. ACM Trans. Auton. Adapt. Syst. 6, 2, Article 15 (June 2011), 26 pages.
DOI = 10.1145/1968513.1968518 http://doi.acm.org/10.1145/1968513.1968518

1. INTRODUCTION

Amorphous Computing [Abelson et al. 2007] considers computational particles dispersed irregularly in an environment and communicating locally with each other. They form what is referred to as an "amorphous computer". Particles are programmed identically, but may store different values. Particles are all similar and generally stationary. Mobile particles considered so far are either swarms of robots or self-assembling robots. Primitives for programming amorphous computing take inspiration from self-organizing natural systems, and corresponding applications show a high level of robustness to particle errors.

This work was partially funded by projects EVE (TIN2009-14702-C02-01), IEA (TIN2006-15662-C02-01), and Agreement Technologies (CSD2007-0022), and by the Generalitat de Catalunya under the grant 2009-SGR-1434.
J. L. Fernandez-Marquez holds a FPI scholarship from the Spanish Government.
Author's addresses: J. L. Fernandez-Marquez (contact author) and J. L. Arcos, IIIA-CSIC, Campus de la UAB, E-08193 Bellaterra, Catalonia, Spain; email: fernandez@iiia.csic.es; G. Di Marzo Serugendo, Birkbeck College, University of London, UK.

DOI 10.1145/1968513.1968518 http://doi.acm.org/10.1145/1968513.1968518

Spatial computing builds on the amorphous computing concept by considering physical geographical zones as computing elements while making abstractions of underlying computational devices (particles such as sensors, mobile phones, robots), for example, geographical zones able to process tasks, collaborate with each other in order to produce some specific result.

Our goal is to provide a spatial memory service for mobile user applications, where both stationary and mobile devices provide memory storage. A spatial memory is a set of (active) geographical zones (of any shape, possibly overlapping) each acting as a memory cell able to store any kind of information. This memory would constitute a base service for the spatial computing paradigm. For instance, a spatial search and rescue service could coordinate an emergency service to rescue survivors of a natural disaster by exploiting data about the survivors' positions and data about rescue team availability, both stored in such a memory. Additionally, data can change while it is replicating (e.g., to create gradient fields) or different pieces of data can self-aggregate to create new data.

The challenge is to provide persistent storage (and retrieval) of information at specific locations on top of a volatile (mobile and uncontrolled) storage media. By persistent, we mean that the data must be stored and be accessible at a fixed geographical area for the duration required by the application (from a few minutes, to several hours, to several days).

Our work so far has concentrated on persistent *storage* algorithms using the concept of hovering information, thus providing a way of implementing the idea of spatial memory in an infrastructure-free and self-organizing way. A piece of hovering information is a geo-localized information residing in a highly dynamic environment such as a mobile ad hoc network. This information is attached to a geographical area, called "an anchor area". A piece of hovering information is responsible for keeping itself alive, available, and accessible to other devices within its anchor area. Hovering information uses mechanisms such as active hopping, replication, and dissemination among mobile nodes to satisfy the above requirements. It does not rely on any central server. The appealing characteristics of the hovering information concept is the absence of a centralised entity and the active participation of the information in the storage and retrieval process. We are also working on *retrieval* algorithms and primitives for a complete spatial memory service, through both simulations and actual implementations using mobile phones. Well-assessed results for retrieval are not yet available and are out of the scope of this article.

Current proposals for location-based services consider data as a passive entity moved around by the infrastructure, intimately linked with the users (publishers or subscribers). Data is either stored on a fixed infrastructure and delivered to users when they reach a certain location [Eugster et al. 2005], or, for infrastructureless scenarios, the publication space moves along with the publisher, and a subscriber space must overlap the publisher space to access the information [Eugster et al. 2009]. Other work for data dissemination over MANETs exploits mobile devices but does not address the persistency of the information or does so for specific scenarios only (e.g., intravehicular networks) [Leontiadis et al. 2009]. Data itself is passive and not self-organizing. Tasks of propagating or routing information are left to the infrastructure (mobile or fixed nodes).

In our work, we consider data as an active self-organizing entity acting as a mobile agent, which decides on its own where to go next and how to be stored. Data has a life of its own; it is dissociated from the (human) users who produce and exploit it, and from the mobile devices who act as storage media. In previous work, we defined the concept of hovering information, provided persistent storage algorithms for circular areas, and investigated performances for single [Villalba Castro et al. 2008] and

multiple (different) pieces of hovering information [Villalba Castro et al. 2009]. Our initial replication algorithms and their corresponding simulations took into account the wireless characteristics of mobile devices and concentrated on circular areas only. We did not investigate other types of shapes, specific issues such as scalability or speed of convergence. Simulations were performed with OMNET++,[1] a simulation tool for wireless devices. Although good for gaining performance results, these simulations did not provide a satisfactory visualization of the propagation of the pieces of information. We decided to revise our algorithms and to undergo additional thorough simulations using Repast,[2] an agent-based simulation tool providing visual simulation and still allowing us to collect performance results.

Therefore, the current article consists of a theoretical investigation of new variants of our persistent self-organizing storage algorithms for any type of shape. In this article, we focus on one single piece of information replicated into a predefined area and consider only the storage aspect of the spatial memory mentioned above. We evaluate algorithm performance by measuring metrics such as number of messages exchanged among mobile nodes, memory consumption, and accessibility rates. Specifically, this article distinguishes itself from our previous articles in the following ways.

— It provides self-organizing storage algorithms for amorphous areas, that is, areas of any shape (not only circular shapes).
— It introduces the notion of repulsion (first introduced by Cheng et al. [2005]), where a piece of information quickly replicates and fills an area without broadcasting to all nodes.
— It provides a visualization of the simulations.
— It discusses convergence and scalability of the algorithms as well as the recovery behavior under the massive failure of nodes.
— Finally, it identifies borderline cases of functioning (e.g., the minimum number of nodes in an area at which the system still works).

Results show that our algorithms are scalable and converge quickly in filling a specific area. Results also show that there is no single algorithm that outperforms the others for all metrics. Variants with repulsion are scalable, but employ more messages to achieve the propagation. Broadcast converges very quickly, but at the cost of all nodes storing the information.

We consider the concept of hovering information as a service: pieces of hovering information come with *policies* specifying the spreading mode (speed and availability level) and the time to live (garbage collection). For instance, a piece of hovering information spreading an emergency message should optimally follow the broadcast algorithm (even if this is more expensive) due to safety issues, since all users must be informed quickly. A piece of hovering information carrying an advertisement message could be cheaper to spread and does not need to be accessible to everybody in the area. Additionally, pieces of hovering information may adapt to the environment. For instance, if using a specific algorithm, the target cannot be reached because if the number of nodes suddenly drops, they can switch to another algorithm in order to maintain the same level of service availability.

The notion of policies allows switching from one algorithm to another: pieces of hovering information use the algorithm that is most appropriate to the type of information (e.g., emergency information vs advertisement), to the propagation mode (faster vs slower) by changing the repulsion rate to the accessibility rate threshold (100% of

[1]http://www.omnetpp.org/
[2]http://repast.sourceforge.net/

the users in the anchor area must have access to the information vs it is acceptable that only 80% of the users do).

The article is organized as follows. Section 2 reports on related work. Section 3 presents the hovering information concepts and the storage areas. A series of storage algorithms are discussed in Section 4. Simulations, measured performance, visualization and analysis of results are reported in Section 5.

2. RELATED WORK

Hovering information and spatial memory are related to different concepts such as memory, middleware, and the dissemination of data. Below, we analyze previous work related to hovering information.

2.1 Location-Based Publish/Subscribe

Publish/subscribe approaches come in different flavors. Because the original publish/subscribe systems were intended for stationary users through Internet access and did not convey the notion of location [Eugster et al. 2003], they have been extended in different ways to accommodate mobility and location. Specifically,

— to accommodate user mobility and location, location-based publish/subscribe was proposed [Eugster et al. 2005]. Information is stored on a fixed infrastructure, and passed on to users when they reach a particular location and their subscription matches a particular published topic;
— to alleviate the need for a fixed infrastructure, proposals exploiting mobile devices led to infrastructureless publish/subscribe, where the publication space moves along with the publisher [Eugster et al. 2009]. Subscribers are notified when their space overlaps the publication space of the publisher.

Hovering information and spatial memory complete this picture by providing a self-organizing infrastructureless memory service for mobile users. Information is stored at some specific geographic area without the need for a fixed infrastructure. It is available to users when they are in this area and is dissociated from its original publisher. We consider this dissociation important because new case studies could be addressed with this new approach. One example of a case study is the presence of ice on the road. One car detects ice and deposits a piece of information with an anchor area center in the ice. The piece of information will be alive and will warn other cars before they reach the ice, while there is at least one car to store and provide the information. In this case study, if we use the location-based publish/subscribe, the car that deposits the information must be there to provide the information to other cars. In our case, the information keeps the position by using other cars to store and provide the information. Even though this article focuses on storage algorithms, we now mention a simple retrieval algorithm (also assumed in the simulations): every node for which there is another node in communication range that holds information has *de facto* access to the information and can retrieve the data (there is no routing implied).

2.2 Dissemination Services

Alternative algorithms for spreading information include opportunistic spatio-temporal dissemination services over MANETs [Leontiadis and Mascolo 2007b], focusing on car traffic-centered applications, epidemic models such as Epcast [Scellato et al. 2005], or gossip models [Datta et al. 2004]. These works provide an interesting starting point for replication algorithms, but do not offer a solution to ensure the persistence of the information.

In the domain of location-driven routing over MANETs, we mention works such as GeoOpss [Leontiadis and Mascolo 2007a]; search and query propagation over social networks like PeopleNet [Motani et al. 2005]; and collaborative services such as collaborative backup of the MoSAIC project [Killijian et al. 2004; Courts et al. 2005]. The work of Leontiadis et al. [2009] provides persistent dissemination of data in intravehicular networks using data that "sticks" to the locations where drivers would receive it.

Geocast [Maihfer 2004] consists in sending information to a group of nodes in a network within a geographical region. Geocast protocols exploit either a fixed or an ad hoc infrastructure and involve routing data from a sender (outside the geographical region) to a group of nodes within the target geographical region. These protocols are concerned with the delivery (routing or forwarding) of messages from a sending to a receiving region. The main goal of hovering information or the spatial memory concept is to store data at some geographical region and to make it available to all nodes in that region. There is no specific routing, the data is produced from the area where it is consumed. The nodes collaboratively serve as storage media.

All these approaches consider the data itself as a passive entity multicasted by mobile devices. Self-organizing data, in the form of mobile agents as proposed in this article, supports spatial computing scenarios better by adapting to local conditions (applying ad hoc replication algorithms), or by modifying itself during the dissemination process.

2.3 Virtual Nodes

The virtual infrastructure project [Dolev et al. 2003, 2004, 2005a, 2005b] aims to set up a set of virtual nodes having a well-known structure and trajectory over a mobile ad hoc network. Virtual nodes are equipped with a clocked automaton machine, useful for implementing distributed algorithms such as leader election, routing, atomic memory, motion coordination, and so on. This approach works on offering a structured abstraction layer of virtual nodes.

Within this project, the GeoQuorum approach [Dolev et al. 2003] proposes the implementation of an atomic shared memory in ad hoc networks, working in two parts. First, mobile hosts populate geographical regions, called focal points, that must not overlap. Within a focal point, mobile hosts cooperate to implement the notion of abstract node. Focal points represent virtual processes, and each focal point is required to support a local broadcast service providing reliable and totally ordered broadcast (all nodes receive the same information and in the same order). Second, the notion of atomic shared memory is then actually built on top of the (static) abstract nodes using a Geocast to communicate with the focal point nodes. The GeoQuorum algorithm ensures fault-tolerance and availability of data by replicating memory data at a number of focal points.

Hovering information takes a different approach, where each piece of hovering information is an autonomous entity responsible for its own survivability by exploiting the dynamics for underlying network purpose. The spatial memory concept is very similar to the atomic shared memory (notion of virtual node and memory). The main difference is that spatial memory is intended to be a best-effort service (data may not be available and fault-tolerance is not specifically addressed), memory cells can overlap, data is active, and decides where to go next.

2.4 Persistent Node

The notion of *PersistentNode* from Beal [2003] is inspired by the amorphous computing paradigm. An amorphous network is a set of uniformly randomly distributed particles over a 2D region. A PersistentNode is a key/value pair residing in a geographic region

(a central particle and a circle of particles around that particle, all holding the same key/value pair). The node (in fact the set of key/value pairs as a circle) may move towards another region of the 2D space (hop from particles to particles), especially when particles in the node are damaged.

As in the case of hovering information, data is the active entity and the storage medium is rather passive regarding the data. A PersistentNode (i.e., a group of data inside some particles) resides at some geographic locality and may move if necessary. However, the storage media, that is, the particles, even though they can fail are not mobile, and all particles in the node store the data. Hovering information considers mobile particles. At the moment, the "node" provided by hovering information is itself fixed and specified at some location. Our algorithms are such that data follows (is attracted by) the center of its storage area. In the case where the storage area moves, the data just follows the center.

2.5 Middleware

The TOTA middleware (Tuples-On-The-Air) [Mamei and Zambonelli 2001, 2005] proposes an API to support the development of adaptive context-aware applications in pervasive and mobile computing scenarios. The main component are tuples, which are propagated through the devices composing the system via the ad hoc network of the infrastructure. TOTA follows a Linda-like approach to store and retrieve tuples through pattern matching. The tuples, defined by the user application, are composed of three attributes: the content, the rules of propagation, and the rules of maintenance. Spatial memory could be implemented on top of TOTA by defining its own tuples and rules of propagation and maintenance.

2.6 Data Clouds

The Hovering Data Clouds (HDC) concept [Fekete et al. 2006; Wegener et al. 2006], which is part of the AutoNomos project, is applied to the design of a distributed infrastructure-free car traffic congestion information system. Although HDCs are defined as information entities that have properties similar to hovering information, the algorithms described do not consider them as an independent service but as part of the traffic congestion algorithms. The hovering information dissemination service is thought of as a service independent of the applications using it. The Ad-Loc system [Corbet and Cutting 2006] is an infrastructure-free location-aware annotation system that shares similarities with hovering information. However, Ad-Loc does not focus on studying properties such as the critical number of nodes or dealing with self-organizing algorithms that allow the information to adapt its behavior according to the network saturation.

2.7 Viral Programming

Paintable computing [Butera 2007] is a type of viral programming which follows the amorphous computing concepts. A display is composed of numerous computing particles (very small fixed nodes) dispersed randomly on a "screen". The programming model for paintable computing is based on the self-assembly of mobile code.

Pieces of hovering information are clearly mobile codes, self-replicating among mobile nodes. Paintable computing considers fixed particles, while in our case the particles are mobile. Pieces of hovering information are not self-assembling, even though we have considered the notion of swarms of pieces of hovering information: a piece too big to get stored into a single node breaks down into smaller pieces stored in different nodes; they later self-assemble to produce the original data [Di Marzo Serugendo et al. 2007].

Smart messages [Borcea et al. 2004] are a type of mobile agent and provide an implementation for spatial programming with mobile computing devices. The idea of spatial memory is very close to the notion of spatial programming: storing and retrieving information on top of an unreliable mobile storage media using spatial references. Smart messages follow a self-routing strategy to find their location.

Paradigms such as Amorphous Computing, Paintable Computing, and PersistentNode share similarities with our work: local-knowledge, self-organization, autonomous behavior of entities, biological inspiration, and mobile code.

The novelty of the hovering information service (and later of spatial memory) resides in the combination of the techniques described, in particular the combination of virtual memory, persistent node (active and moving data), and viral programming (mobile code).

3. HOVERING INFORMATION CONCEPT

3.1 Hovering Information

A piece of Hovering Information h is a geo-localized information attached to a geographical area, called the *anchor area*. The main goal of h is to self-replicate among neighboring mobile nodes in order to maintain itself in the specified anchor area and make itself accessible to mobile nodes in that area.

A piece of hovering information h is defined as a tuple:

$$h = (id, A, n, data, policies, size);$$

where id is the hovering information identifier, A is the anchor area (see below), n is the mobile node where h is currently hosted, $data$ is the actual data carried by h, $policies$ are the spreading policies of h, and $size$ is its size.

In this article, we do not investigate the active usage of policies for enhancing adaptation. The policy is the dissemination algorithm applied by the pieces when they spread in their environment. For the sake of completeness, we give the full definition of h (including policies and size), and as we said above, policies can be used to dynamically change the algorithm in runtime.

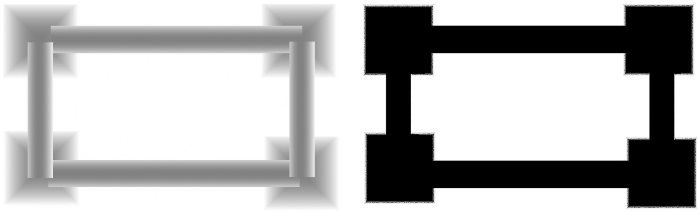
3.2 Anchor Areas

Hovering information spread into indoor and outdoor spaces such as motorways, pedestrian roads, shopping centers, and leisure areas. The shape of the area can vary from a simple circle centered on a focal point to more elaborate shapes of any type (regular, irregular, convex, or not, etc.).

Anchor areas in this article do not have any restriction on the shape, as in previous work [Di Marzo Serugendo et al. 2007] (where anchor areas are circular); we call these amorphous areas. The goal of a piece of hovering information is to spread itself in the area in order to be accessible to nodes in that area (i.e., a piece of information needs to know the anchor area).

Figure 1 represents an anchor area with 4 halls connected by 4 corridors. Nodes can move freely in whatever direction (also outside the corridors), that is, the hall and corridors are not delimited by walls. The information must fill the anchor area, remain located inside it, and must not spread outside. In particular, if a node is located at the center of Figure 1 (thus not in a corridor), it may have access to a replica but should not store one.

This is a typical area for a shopping center and for a piece of information that must be relevant to people located in the corridors and meeting points, lifts, or stairs (4 corners), but not when they are inside specific shops.



(a) Gradient (b) Binary

Fig. 1. Amorphous areas.

3.2.1 Binary Matrix and Gradient Matrix. The amorphous areas (i.e., anchor areas without a regular shape) are implemented using a matrix. A matrix is a discretization of the real space where the nodes reside (i.e., the environment). A piece of information uses the matrix to know the position of the host inside an amorphous area. Given a host position, the binary matrix (Figure 1(b)) denotes only whether or not a position is inside the amorphous area. Moreover, the gradient matrix (Figure 1(a)) adds gradients to denote the relevance inside the anchor area, thus providing a way to coordinate the momevents of the information to more relevant positions inside the amorphous area. Thus, the binary matrix is defined by ($matrix \subset \mathbb{E}\, x\, \{0, 1\}$) where 0 represents positions outside of the area and 1 represents positions inside the area. The gradient matrix is defined by ($matrix \subset \mathbb{E}\, x\, \mathbb{N}$), where $\mathbb{E}$ is the whole environment that is, the set of all geographic coordinates where mobile nodes can move; those closer to the inside of the area have a higher value than those closer to the border.

The matrix has the size of the space of the area we consider. For instance, let us consider a museum area of 300m x 300m. The grid size is 300 x 300: one point in the matrix refers to one meter in the real world. For the binary approach, each position in the matrix contains the information bit 0 or 1. A piece of information, which wants to know if it is in the anchor area, checks its position and looks in the matrix to know if its position is inside the area (1) or outside (0). For the gradient area, we use a matrix too, but now the information in each point is a byte. The 0 value refers to the area outside the anchor area. The value increases when the replica is inside the anchor area, 255[3] being the inner part of the anchor area (a darker shade in the pictures). Thus, the center of the corridor has a value of 255 as does the center of the hall.

The actual set of coordinates that is in an amorphous area is given by

$$A(matrix) = \{b \in \mathbb{E} \mid ((b, val) \in matrix \wedge val \neq 0)\}.$$

We consider that the overhead for storing the matrix is neglictible compared to the actual data (a few bits for each square meter).

3.3 Assumptions

Pieces of hovering information have the following information (at any point in time t):

— knowledge of the anchor area (either a pair $A = (a, r)$, anchor location and radius; or a binary or gradient matrix);
— position of the node it is currently in;
— position of neighboring nodes and those that have the information already.

[3]this could be less: 4, 8, 16, ...)

Algorithm 1: Broadcast Replication Algorithm

```
pos ← NodePosition()
if (IsInAnchorArea(pos)) then
    Broadcast()
end
```

Algorithm 2: Attractor Point Replication Algorithm

```
pos ← NodePosition()
neigbourNodes ← NodeNeigbours()

if (IsInAnchorArea(pos)) then
    selectNodes ← SelectKrHighestGradient(neigbourNodes)
    Multicast(selectNodes)
end
```

We make the assumption that the information itself is more expensive to spread around than getting information about position and data id stored by neighboring nodes. Due to the dynamic nature of the nodes, a hovering information service provides a best-effort service accommodating imprecise positions or the unexpected movement of nodes.

4. ALGORITHMS

In this section we present the various replication algorithms that we compare in this article.

4.1 Broadcast

At every simulation step (every second of simulation time), each piece of information executes the Broadcast replication algorithm. In a real implementation, pieces of hovering information would apply the algorithm at a regular interval (e.g., every second), but not in a synchronous way, as it is for the simulations. In the Broadcast algorithm, the replication is triggered when the information is inside the area (Algorithm 1). Broadcast replicates to all nodes in communication range that do not hold a replica. This can be viewed as a kind of multicast; we prefer to call it broadcast because the effect is to put a replica in each node.

Broadcast spreads data in the neighborhood (to those nodes that do not yet store the information). There is no selection of a subset of nodes based on their geographical position, as in geocast.

4.2 Attractor Point

The Attractor Point algorithm avoids broadcasting to all neighboring nodes. At each simulation step, a piece of information replicates only to the *Kr* nodes in the communication range that present the highest level of gradients. The Attractor Point algorithm does not apply to a binary matrix. The algorithm is triggered when the information is inside the area (Algorithm 2).

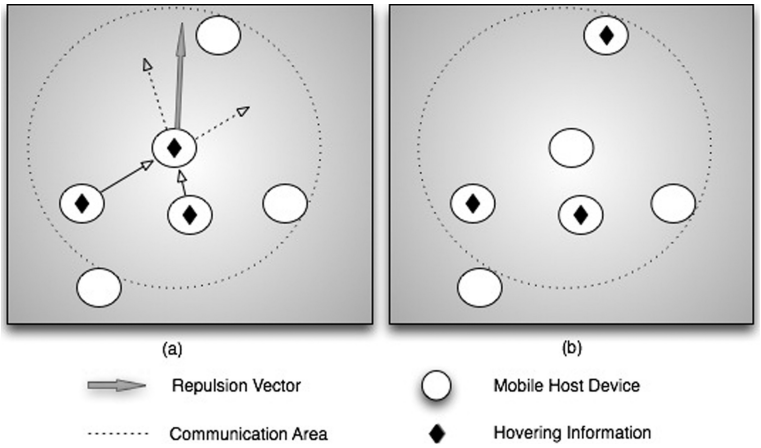


Fig. 2. Repulsion.

4.3 Cleaning

A piece of hovering information located outside of the amorphous area (anchor area with any shape) removes itself and frees the memory of the node in which it was stored. The cleaning algorithm ensures that the hovering information pieces stay in the anchor area and do not spread all over the environment. Additionally, we consider that at most one replica of the same piece of hovering information is stored in a given node at a certain point in time.

4.4 Repulsion

The repulsion mechanism helps the system to spread the pieces of information over the anchor area, maintaining good accessibility levels while keeping a minimum number of replicas (in order to use less memory). If two or more replicas are close to each other, one of them will move away (remove itself from its current location and replicate further away). Additionally, repulsion enables to easily fill amorphous shapes: the information spreads along the shape until it fills it. The repulsion mechanism, inspired by gas theory, has been used in self-repairing formation for swarms of mobile agents [Cheng et al. 2005] and as an exploration mechanism in multiswarm optimization algorithms [Fernandez-Marquez and Arcos 2009]. The main difference between our system and the self-repairing formation for swarms of mobile agents is that pieces of hovering information do not have any control over the movement of the mobile nodes.

Figure 2(a) shows how a replica creates a repulsion vector inversely proportional to the distance between itself and neighboring replicas, and how this replica subsequently moves to another node following the repulsion vector, as we see in Figure 2(b).

Contrary to the previously described Broadcast and Attractor Point algorithms, the replica, which applies the repulsion mechanism, removes itself from the current node.

Let h be a piece of information, r a replica of h, and $n(r)$ the mobile node where r is currently located. Using the repulsion mechanism, the desired position for r at time $t+1$, $\vec{P}_d(r)_{t+1}$, is calculated as follows:

$$\vec{P}_d(r)_{t+1} = \vec{P}(r)_t + \vec{R}(r)_t, \tag{1}$$

Algorithm 3: Broadcast with Repulsion - Binary Matrix

```
pos ← NodePosition()
neigbourNodes ← NodeNeigbours()
if (IsInAnchorArea(pos)) then
    CalculateDesirePositionbyRepulsion() equations (1), (2)
    MoveToNodeClosestDesirePosition()
    if (¬ExistPieceOfInformation(neigbourNodes)) then
        Broadcast()
    end
end
```

where $\vec{P}(r)_t$ is the position of r at time t and $\vec{R}(r)_t$ is the repulsion vector at time t.

$$\vec{R}(r)_t = \sum_{i \in R(r,t)} \frac{\vec{P}(n(r))_t - \vec{P}(n(i))_t}{dist(n(r), n(i))} \times (r_{comm}(n(r)) - dist(n(r), n(i))), \tag{2}$$

where

— $R(r, t)$ is the set of replicas of h in communication range of $n(r)$ at time t;
— $dist(n(r), n(i))$ is the Euclidean distance between the node $n(r)$ and the node $n(i)$;
— $\vec{P}(n(j))_t$ is the position of node $n(j)$ where the replica j is stored at time t;
— $\times$ is the multiplier operator; and
— r_{comm} is the communication range.

Once the desired position $\vec{P}_d(r)_{t+1}$ is known, the replica r must choose which node in its communication range is the closest to this desired position. If the closest node is itself, then repulsion is not applied. Otherwise, r replicates to the new node and deletes itself from $n(r)$. Basically, these equations produce a repulsion vector that moves the replica of information to the less dense area inside its communication range.

4.4.1 Broadcast Repulsion. The broadcast with Repulsion algorithm (see Algorithm 3) uses broadcast as a replication algorithm and repulsion to spread the replicas away. A piece of information replicates to all the neighbors when it is inside the anchor area and there are no other replicas in the communication range. Moreover, a piece of information executes repulsion when it is inside the area and there are one or more replicas in the communication range. Thus, replication with broadcast creates the replicas necessary to cover the shape and repulsion is responsible for distributing the pieces of information uniformly over the area. Broadcast repulsion is applied to the binary matrix, since broadcast replicates to all the neighbors and does not use the gradient information.

The main steps of the algorithm are shown in Figure 3. At the beginning, Step 1, the first piece of information is created and the anchor area is defined. This piece of information executes the broadcast, since there is no other piece of information in its communication range. Step 2 shows how the nodes that are in the communication range get a copy of the piece of information. At Step 3, the pieces of information spread due to the Repulsion algorithm. After the repulsion step, two of the three replicas have no other replica in their communication range, they then execute the broadcast algorithm and replicate to all the nodes in their communication range, Step 4. At Step 5, the pieces of information spread further due to repulsion, and finally fill the anchor area. Step 6 shows how, without getting stored in every node, the initial

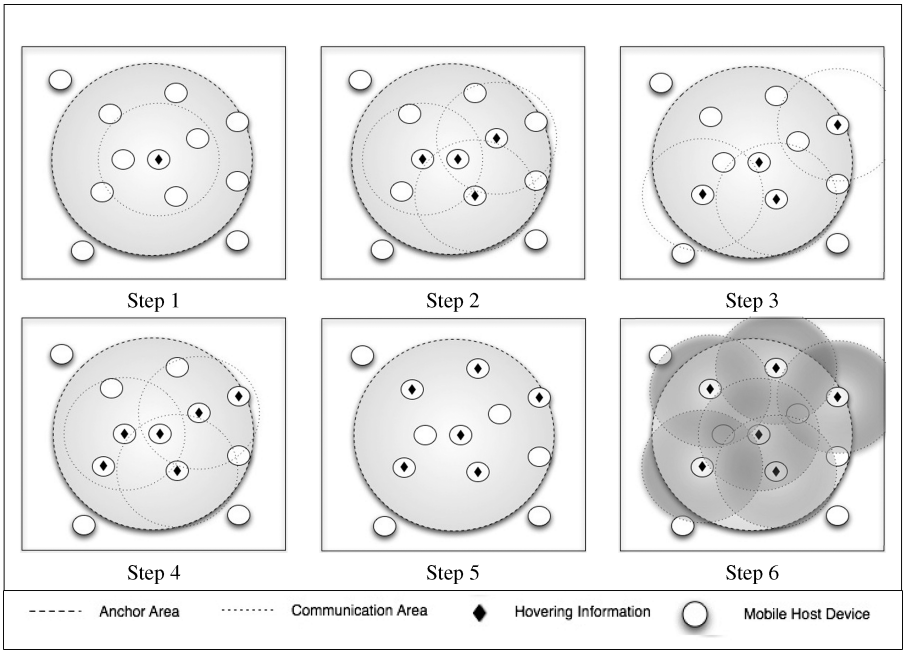


Fig. 3. Broadcast repulsion steps.

piece of information becomes available to all nodes in the anchor area (i.e., it "fills" or "covers" the whole anchor area).

4.4.2 Attractor Point Repulsion. The attractor point with repulsion algorithm (see Algorithm 4) uses multicast as a replication algorithm and, analogously to broadcast repulsion, repulsion to spread the replicas away. A piece of information executes muticast to the Kr neighboring nodes with a higher level of gradient when it is inside the anchor area and there are not other replicas in the communication range. The use of the gradient provides a better location for the new replicas created. Analogously to the broadcast repulsion algorithm, a piece of information executes repulsion when it is in the anchor area and there are one or more replicas in the communication range. Then, algorithm behavior is similar to broadcast repulsion, but attractor point repulsion reduces the number of replicas created in each replication process. Attractor point repulsion requires the gradient matrix to choose the Kr nodes with the higher level of gradient.

4.5 Metrics

We use the following notation: $HoverInfo_t = (\mathcal{N}_t, \mathcal{H}_t)$ is a hovering information system at time t. $\mathcal{N}_t$ is the set of nodes present in the system at time t. $\mathcal{H}_t$ is the set of pieces of hovering information and any replica present in the system at time t. Since we consider only one piece of information in this article, $\mathcal{H}_t$ stands for h and all its replicas (same identity and data but stored at different nodes). Note that h itself is just a specific replica. We measure and compare the performances of each algorithms against the metrics below.

4.5.1 Survivability. A piece of hovering information is alive at some time t if there is at least one node hosting a replica r of this information.

Algorithm 4: Attractor Point with Repulsion - Gradient Matrix

```
pos ← NodePosition()
neigbourNodes ← NodeNeigbours()
if (IsInAnchorArea(pos)) then
    CalculateDesirePositionbyRepulsion() equations (1), (2)
    MoveToNodeClosestDesirePosition()
    if (¬ExistPieceOfInformation(neigbourNodes)) then
        selectNodes ← SelectKrHighestGradient(neigbourNodes)
        Multicast(selectNodes)
    end
end
```

Definition 1. *Survivability of Hovering Information h at time t.* The survivability of h at time t is given by the boolean value:

$$sv(t) = \begin{cases} 1 & \text{if } \exists r \in \mathcal{H}_t, n(r) \in \mathcal{N}_t \\ 0 & \text{otherwise.} \end{cases}$$

Survivability along a period of time is defined as the ratio between the amount of time during which the hovering information has been alive and the overall duration of the observation.

Definition 2. *Rate of Survivability of Hovering Information h at time t.* The survivability of h between time t_c (creation time of h, always 0 in this article) and time t is given by

$$SV(t) = \frac{1}{t - t_c} \sum_{\tau=t_c}^{t} sv(\tau).$$

4.5.2 Accessibility. A piece of hovering information h is accessible by a node n at some time t if the node is able to get this information, that is, if there exists a node m in the communication range of the interested node n and which contains a replica of h.

Definition 3. *Accessibility of Hovering Information h for node n at time t.* Let $n \in \mathcal{N}_t$ be a mobile node, the accessibility of h for n at time t is given by the boolean value:

$$ac(n,t) = \begin{cases} 1 & \text{if} : \exists r \in \mathcal{H}_t, n(r) : \text{in } A_C(n) \\ 0 & \text{otherwise.} \end{cases}$$

$A_C(n)$ is the area in the communication range of node n.

Definition 4. *Accessibility of Hovering Information h at time t.* The accessibility of h at time t is given by

$$AC(t) = \frac{S\left(\bigcup_{r \in \mathcal{H}_t} A_C(n(r)) \cap A\right)}{S(A)},$$

where $S(X)$ denotes the surface area of X, $A_C(n(r))$ is the area of communication of the node n storing replica r, and A is the anchor area.

4.5.3 Messages. A message is sent each time a replica self-replicates to another node.

Definition 5. *Number of messages sent at time t.* Let $msg(n,t)$ be the number of messages sent by node n between 0 and t, the number of messages sent at time t is given by

$$MSG(t) = \sum_{n \in \mathcal{N}_t} msg(n,t).$$

4.5.4 Memory. The memory that the system uses at time t is the total number of replicas in the system at time t: $|\mathcal{H}_t|$.

Definition 6. *Rate of Memory used at time t.* The rate of memory used at time t is

$$MEM(t) = \frac{1}{t - t_c} \sum_{\tau=t_c}^{t} |\mathcal{H}_\tau|.$$

For the simulations and the metrics above, we used a synchronous model. In an actual implementation, we would not measure those metrics on-the-fly, but we would measure the metrics above by analyzing mobile device activity (e.g., how many times they received, replicated, and discarded data) during a specific period of time.

5. SIMULATION RESULTS

We conducted diverse simulations involving the algorithms described above for different scenarios. We used the Recursive Porous Agent Simulation Toolkit (REPAST),[4] a Java environment for agent simulations. In addition to performance measures, it provides an event scheduler for simulating concurrency and a two-dimensional agent environment that we use to visualize the nodes and the spread of pieces of hovering information in the environment.

We noticed that broadcast with a gradient matrix behaves in the same way as broadcast with a binary matrix (broadcast applies as long as a piece is inside the area, independently of the gradient). Attractor point makes sense for a gradient matrix only, since replicas are attracted to the center of the area (the highest values in the matrix). We do not consider the attractor point for a binary matrix.

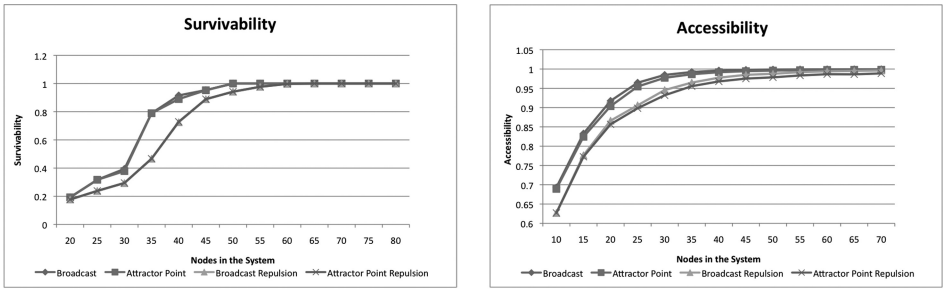
5.1 Anchor Areas

Although the WLAN 802.11 standard provides communication ranges up to 70m for indoors and 250m for outdoors, most authors (e.g., Roth [2003]) consider that this is too high for realistic situations, and propose smaller values. We defined two scenarios: The *indoor scenario* where the communication range is 40 meters and the *outdoor scenario* where the communication range is 80 meters. The anchor area is preseted in Figure 1, and the mobility model used for the nodes is the Random Way Point. This mobility pattern, in contrast to other patterns like random direction or random walk, creates the highest density of nodes at the center of the environment. By locating the anchor area at the center, a similar concentration of nodes across our different experiments is ensured, that is, we can infer meaningful conclusions. The goal of the simulations is to compare the Attractor Point (AP), Broadcast (BB), Attractor Point Repulsion (APR), and the Broadcast Repulsion (BBR) algorithms in the different scenarios, where the goal is to ensure high accessibility levels with a minimum number of messages and a minimum number of nodes actually storing the information.

[4]http://repast.sourceforge.net/

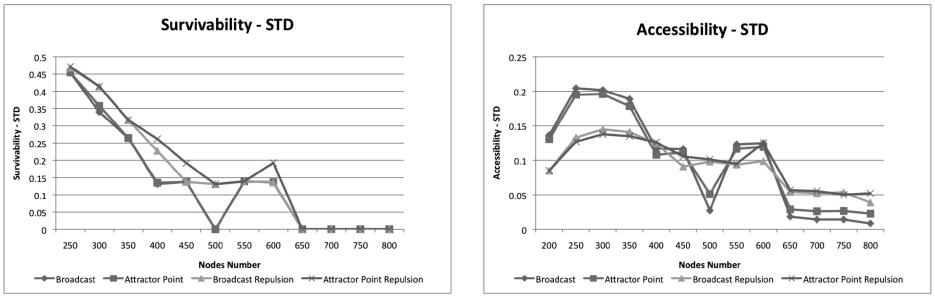
Table I. Scenario's Settings

Blackboard	1200m x 700m
Mobility Model	Random Way Point
Nodes speed	1m/s to 2m/s
Communication Range	40m or 80m
Replication Time (T_R)	1s
Repulsion Time	10 s
Cleaning Time (T_C)	1s



(a) (b)

Fig. 4. Survivability and accessibility: Indoor scenario.



(a) (b)

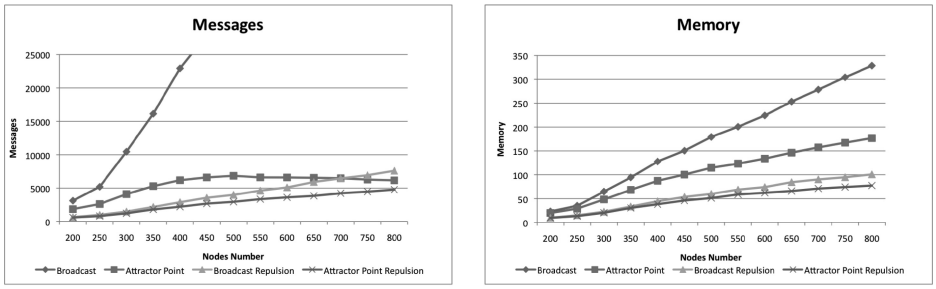
Fig. 5. Survivability and accessibility STD: Indoor scenario.

Table I summarizes the parameter settings for the scenarios. For each experiment, we executed 50 runs, each run lasting 1000 simulation seconds. The results are the average over these 50 runs.

5.1.1 Indoor Scenario. In the indoor scenario, the communication range of the nodes was set to 40m. The anchor area is the one shown in Figure 1 (4 corridors and 4 corners), localized in a bidimensional space of 1200m by 700m and the minimum number of nodes is above 200.

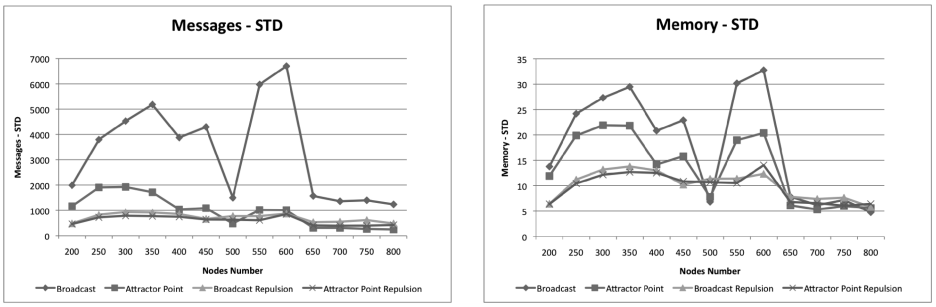
Figure 4 shows survivability and accessibility rates of the four algorithms in this section. Again, Broadcast and Attractor Point algorithms outperform their respective variants with repulsion because more nodes store the data.

Figure 5(a) shows the standard deviation (STD) related to survivability. We observe how STD goes down when the number of nodes increases. That is, the algorithm reduces the probability of failure when the number of nodes increases. When the number of nodes is higher than 600, the STD tends to 0., that is, at least 600 nodes are



(a) (b)

Fig. 6. Messages and memory: Indoor scenario.



(a) (b)

Fig. 7. Messages and memory STD: Indoor scenario.

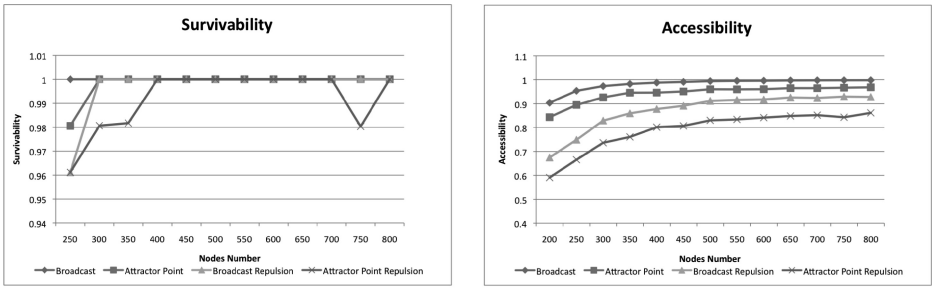
necessary to ensure the survivability of the hovering information along the simulation. Figure 5(b) shows the STD related to accessibility. The best STD is achieved when the number of nodes in the system is enough to ensure the survivability of the hovering information during the entire simulation. Notice that in both of the figures an accidental increase of the STD at 550 and 600 nodes occurred because one of the 50 runs failed to reach survivability. This effect is also visible in Figure 4(a)—a small drop in survivability is visible for 550 and 600 nodes.

Figure 6(a) shows that Broadcast uses many more messages and much more storage memory than the other algorithms. It is clearly outperformed by the rest.

Figure 7(a) shows the STD of messages achieved by the different algorithms in the amorphous area. Except for Broadcast, which presents a high STD, the rest of the algorithms present a STD lower than 10%. Figure 7(b) shows the STD related to memory. All the algorithms reach a STD for memory lower than 8% when the system has the number of nodes to sustain the data. Here again, we observe that the same runs that affected the STD in Figures 5(a) and (b) also affect Figures 7(a) and (b) for 550 and 600 nodes. As in all the STD graphs in this work, a low number of nodes involves an unstable survivability, producing an increment in the STD.

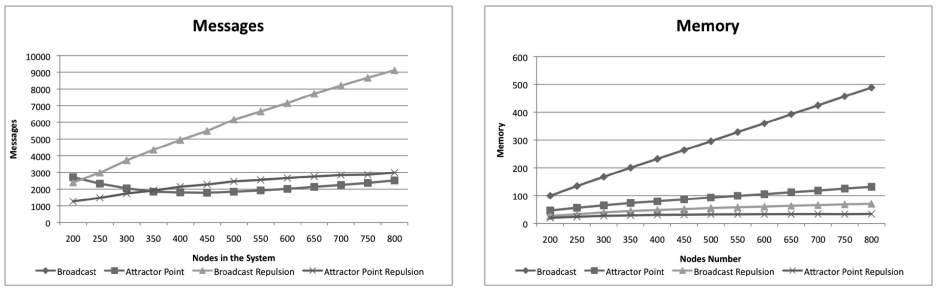
Surprisingly, the Attractor Point with Repulsion algorithm outperforms its variant without repulsion for both the number of messages and memory storage. This result is motivated by the fact that the repulsion variant replicates only when there is no other replica in the communication range.

5.1.2 Outdoor Scenario. The outdoor scenario uses the Random Way Point mobility model and a communication range of 80 meters.



(a) (b)

Fig. 8. Survivability and accessibility: Outdoor scenario.



(a) (b)

Fig. 9. Messages and memory: Outdoor scenario.

Figure 8 shows survivability and accessibility rates for the outdoor scenario. Due to the increased communication range, both rates are better for all algorithms. The Broadcast and Attractor Point still outperform (even though slightly) the repulsion variants for accessibility rates.

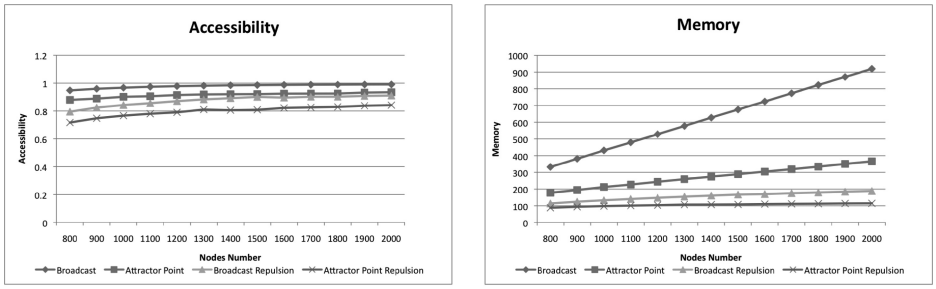
Analogously to the indoor scenario, Broadcast is clearly outperformed by the other three algorithms for both the number of messages sent and the memory used. To facilitate understanding, the messages for Broadcast are not included in Figure 9. Attractor Point (both variants) show similar levels of messages, whereas the repulsion variant uses less memory.

5.1.3 Visualizations. In the Appendix, we provide some simulation images of the indoor scenario using anchor area 1 for the Attractor Point with Repulsion (Figure 20) and the Broadcast algorithm (Figure 22). Moreover, we provide some simulation images for these algorithms when the nodes fail in a subarea (Figure 21 and Figure 23). The images show faster refill after a failure.

We can see how the convergence speeds vary for these algorithms. The Attractor Point with Repulsion takes more time to spread replicas over the whole area (280 seconds of simulation time), while Broadcast fills the shape quickly (at the 18th simulation second). The difference in the number of nodes storing a replica (darker dots) is also clearly visible.

We experimented with massive failures of nodes (Steps 4 to 6); these will be explained later.

We experimented with different shapes, as for instance an anchor area taking the shape of the number five (Figure 24). The spread of information for this area is given in Figure 26.



(a) (b)

Fig. 10. Accessibility and memory: Scalability indoor scenario.

5.1.4 Conclusions. For both indoor and outdoor anchor areas, Attractor Point (both versions) is clearly the best option. Specifically, for higher availability rates (above 95%) Attractor Point is preferable, whereas for acceptable lower availability rates (80%) and less memory storage, Attractor Point with Repulsion is the best option. A full version of this article with more simulations can be found in the technical report [Fernandez-Marquez et al. 2010].

5.2 Analysis of Algorithms

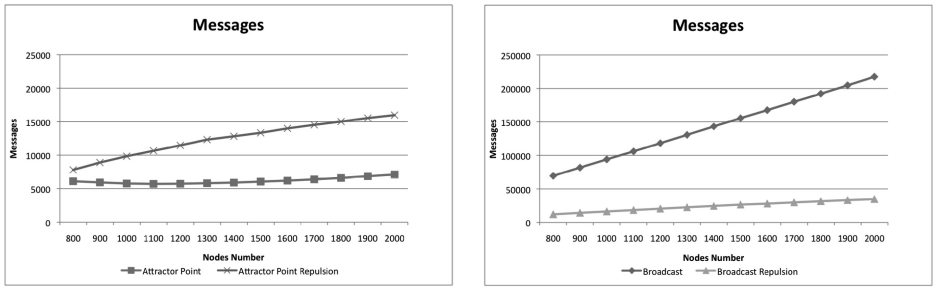
This section investigates scalability issues, the impact of the repulsion rate on the results, in cases where a newly created piece of information does not succeed in spreading and dies almost immediately because the system cannot work below a minimum number of nodes, and a recovering scenario in case of a massive failures of nodes. A more extensive study of failures can be found in Fernandez-Marquez et al. [2010], where the focus is on the failure tolerance of the proposed algorithms in a similar anchor area and parameter setting.

5.2.1 Scalability. In this section we study the behavior of the algorithms for very large numbers of nodes (up to 2000). This experiment involves the anchor area of Figure 1 and a communication range of 40m. We varied the number of nodes from 800 to 2000. Analogously to the other experiments, the Attractor Point and Attractor Point with Repulsion algorithms use the gradient matrix (Figure 1(a)), while the Broadcast and Broadcast with Repulsion use the binary matrix (Figure 1(b)).

The four algorithms achieve very good accessibility rates due to the very large number nodes involved, as shown in Figure 10(a). Broadcast and Attractor point present a better accessibility than the variants with repulsion, since they use also more memory, as shown in Figure 10(b). The main point is that the repulsion variants are scalable regarding memory consumption. The memory levels remain constant despite the higher number of nodes.

The main drawback of applying repulsion is the high number of messages sent when compared to the variants without repulsion. Attractor Point (without repulsion) is scalable in terms of messages: the number of messages remains constant even though the number of nodes increases Figure 11(a). Broadcast (both variants) is not scalable at all in terms of messages Figure 11(b).

To conclude: for a large number of nodes, Attractor Point scales better in terms of messages than the Attractor Point with Repulsion, but employs more memory (up to four times more). The repulsion variant scales in terms of memory but uses far more messages. It is also worth noting that the Attractor Point doesn't use any mechanism to spread the information over the area. The area gets filled as a result of the



(a) (b)

Fig. 11. Messages - Scalability in indoor scenario.

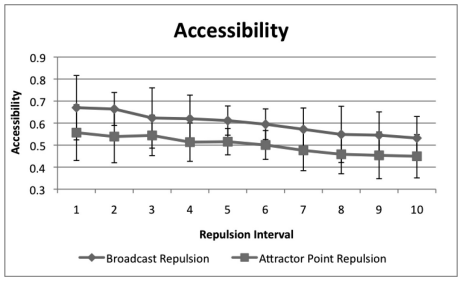


Fig. 12. Accessibility: Varying the repulsion interval.

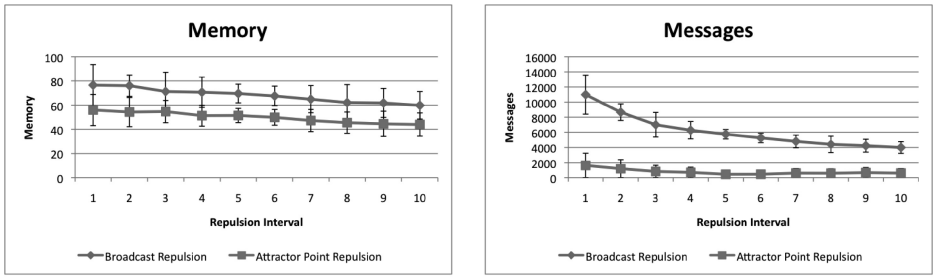
movements of nodes. Nodes help the algorithm in spreading the information. The repulsion version ensures that the information spreads over the area, even when the nodes are stationary or less mobile. Broadcast (without repulsion) is not scalable and its variant with repulsion should be preferred.

5.2.2 Repulsion Interval. An important parameter of the repulsion mechanism is the repulsion interval, that is, the time between two repulsion executions. When the repulsion interval is shorter, the number of messages and the accessibility rate increase. When the repulsion interval is long, the number of messages decreases, but also the accessibility rate.

In this experiment we computed the accessibility rate, the number of messages, and the memory usage along 1000 simulation steps over the average of 50 runs for Broadcast with Repulsion and Attractor Point with Repulsion. For this experiment we set the number of nodes to 500 and the communication range to 40m; we considered the same anchor area as previously.

Figure 12 shows that accessibility rates are better with a shorter repulsion interval (every 1s or 2s), since repulsion spreads the pieces of information over the area faster. Repulsion adapts quickly to topology changes by moving the replicas around. When we increment the repulsion interval, repulsion applies less frequently (every 4s or less) and adapts less quickly to topology changes, and accessibility rates decrease. For memory consumption, a shorter repulsion interval causes more nodes to store replicas, while a longer repulsion interval decreases the memory used. A shorter repulsion rate increases the number of time repulsion applies, that is, improves the exploration of the space and is able to find better places where to replicate, increasing accessibility. The drawback is the higher memory consumptions, as shown in Figure 13.

The main issue with a shorter repulsion interval is that the increase in the number of messages is not proportional to the accessibility rate. In Figure 13(b), we observe



(a) (b)

Fig. 13. Memory and messages: Varying the repulsion interval.

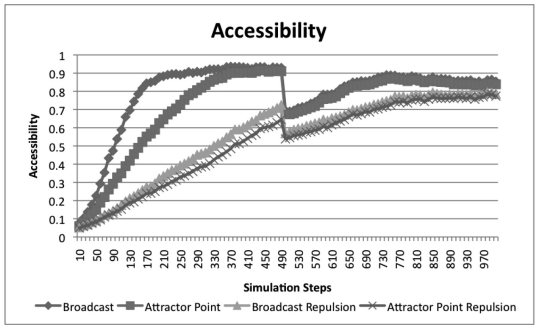
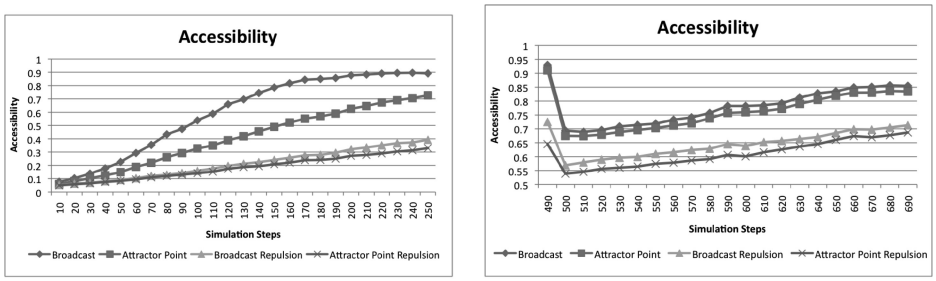


Fig. 14. Accessibility (steps 0 to 1000): Indoor scenario.

that, for short repulsion intervals, the number of messages increases, but this increment is not linear like the accessibility or the memory values. The gain in accessibility is weak compared to the large number of additional messages needed to reach that level. Additionally, the repulsion interval depends on the dynamics of the network. When the network is very dynamic, that is, the topology is continuously changing, a short repulsion interval allows keeping up with the change. For less dynamic networks, a short repulsion interval causes replicas to move around continuously without improving the performance.

5.2.3 Convergence. In this section our goal is to analyze the velocity of convergence when covering the whole area for each algorithm. We first measure the convergence at the initialization and then we investigate the self-healing property of the system when, due to an extreme case of node failure, a large number of nodes are disconnected all at once, and the system has to converge again to fill the area. Simulations run over the anchor area shown in Figure 1. We set the number of nodes to 500 and measured the accessibility at different simulation steps. The executions ran over 1000 simulations steps, and at the 500th simulation step, nodes in a portion of the area were disconnected manually, forcing the system to converge again. We performed two sets of simulations, one for the indoor and one for the outdoor scenarios. We consider that the system has converged in covering the whole area when the accessibility rate is bigger than 0.8; that is, when 80% or more of the nodes in the anchor area have access to the information.

A visual view of the convergence speed of Attractor Point with Repulsion and Broadcast is provided in Figures 20, 21, 22, and 23 respectively, show the disconnected nodes and the recovery phase.



(a) Steps 0 - 250 (b) Steps 490 - 700

Fig. 15. Accessibility with zoom: Indoor scenario.

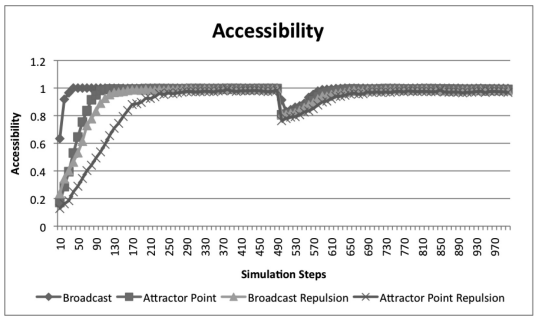


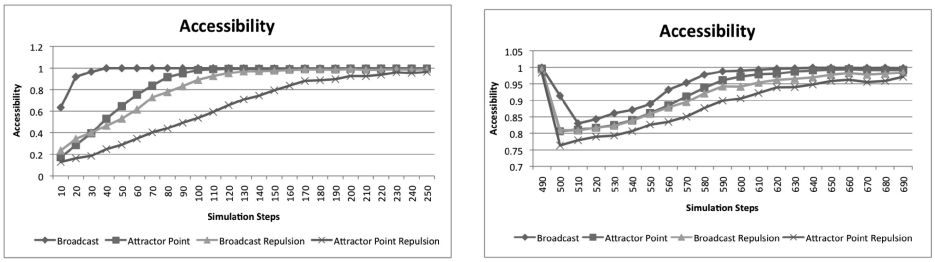
Fig. 16. Accessibility (steps 0 to 1000): Outdoor scenario.

Indoor Scenario. Figure 14 shows that Broadcast provides the best convergence speed. This algorithm reaches the accessibility of 0.8 at the 150th simulation step. This very fast convergence is due to the fact that the information replicates to all the nodes, and thus the area gets filled very quickly. The cost is the high level of messages and memory usage (Figure 6). Attractor Point reaches the accessibility of 0.8 at the 280th simulation step, (see Figure 14). Broadcast with Repulsion and Attractor Point with Repulsion need more than 500 simulation steps to reach an accessibility of 0.8.

At the 500th simulation step, nodes in a portion of the area are disconnected and the accessibility suddenly drops. The speed of convergence after the failure is similar for the four algorithms. As we see in Figure 15(b), the four algorithms follow the same slope, due to the time that the MANET takes to fix the network after the failure, that is, the time required to repopulate the area. Whatever the speed of convergence of the algorithm, after a failure in the network, the speed of convergence is limited by the speed of the nodes in repopulating the area, that is, the speed of convergence decreases due to the lack of nodes to fill the shape.

Outdoor Scenario. With a communication range of 80 meters, the convergence speed increases. Figure 16(a) shows that the Broadcast algorithm reaches 80% of accessibility at the 8th simulation step, Attractor Point at the 65th, Broadcast with Repulsion at the 85th and Attractor Point with Repulsion at 150th. The more memory an algorithm consumes, the quicker it converges. When the failure occurs (Figure 17(b)), we can observe that the convergence speed is similar (same slopes).

5.2.4 Faults—Initialization Phase. One of the key issues is the initialization of the information, that is, the period of time between the creation of a piece of information and



(a) Steps 0 - 250 (b) Steps 490 - 700

Fig. 17. Accessibility with zoom: Outdoor scenario.

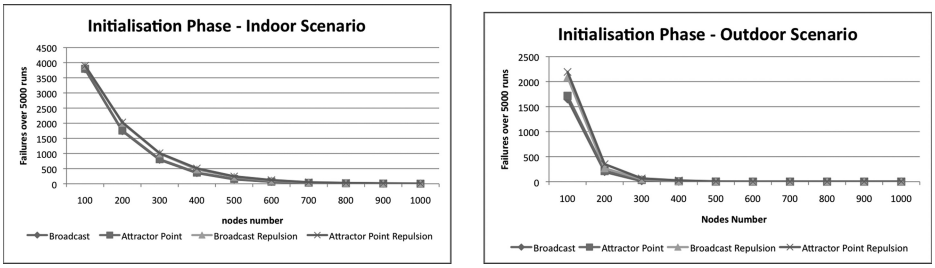
Table II. Init Test for Indoor and Outdoor Scenarios (amorphous)

	Indoor Scenario				Outdoor Scenario			
Nodes Number	BB	AP	BR	APR	BB	AP	BR	APR
100	3780	3799	3897	3900	1646	1716	2088	2194
200	1738	1757	2009	2028	195	213	267	350
300	791	811	998	1012	12	34	48	65
400	354	368	472	509	0	7	9	24
500	147	160	226	244	0	1	0	5
600	64	69	102	123	0	0	1	1
700	27	32	37	50	0	0	0	2
800	14	16	21	27	0	0	0	1
900	5	6	11	13	0	0	0	2
1000	2	2	6	7	0	0	0	3

the moment it covers the whole area. In many cases, due to a lack of nodes in the neighborhood or to random movement, the piece of information *cannot* replicate and dies before the anchor area is fully covered. We observed that once the initialization phase succeeds, that is, once a large part of the anchor area is covered with replicas, the system gains in robustness. Indeed, the probability that *all* the replicas leave the anchor area without replicating to neighboring nodes is lower than during the initialization process (when only one or few replicas are available). During the initialization phase, the system is very fragile and sensitive to random movements.

In this experiment we executed each algorithm 5000 times and each run spent 500 simulation seconds, that is, enough time to ensure that the information was initialized successfully. Over the 5000 runs, we counted the number of times the information died before the end of the run, corresponding to system failure. We studied the four algorithms: Broadcast (BB), Attractor Point (AP), Broadcast with Repulsion (BR), and Attractor Point with Repulsion (APR). The number of failures is shown in Table II (indoor and outdoor scenarios) and Figures 18(a) and 18(b). (Please note that Figures 19 to 26 appear in the online Appendix only, and may be accessed at the ACM Digital Library.)

The indoor case is clearly more sensitive to initial conditions than the outdoor case due to the shorter communication range. The likelihood that random conditions prevent a piece of information from replicating to neighboring nodes is higher in indoor scenarios. In the outdoor case, the repulsion variants fail to initialize more frequently than their counterparts without repulsion. In the indoors case, failures are comparable among the four algorithms. Finally, the more nodes in the environment, the less the risk of failure during the initialization phase.



(a) Indoor (b) Outdoor

Fig. 18. Faults: Initialization phase.

Table III. Node Limit in Indoor and Outdoor Scenarios

	Indoor Scenario		Outdoor Scenario	
Algorithm	Nodes Number	Std. Dev.	Nodes Number	Std. Dev.
BB	167.63	7.86	93.83	5.12
AP	169.02	7.86	98.92	5.02
BR	185.83	10.35	111.98	6.61
APR	186.20	10.63	116.18	6.79

Broadcast and Attractor Point ensure a good survivability rate, they also give a better convergence speed. The price is a higher level of memory storage. Broadcast with Repulsion and Attractor Point with Repulsion provide a very low use of memory. For this reason, the best option is to use algorithms like Broadcast or Attractor Point during the initialization of the system. Later, the information could switch to Broadcast with Repulsion or Attractor Point Repulsion in order to keep the use of memory low.

5.2.5 Faults—Critical Mass of Nodes. We focus here on analyzing the minimum number of nodes that the system needs in order to keep the information alive (survivable) during the whole simulation time. In this experiment we started with a large number of nodes (600) in order to ensure the correct initialization of the system (and to avoid the initialization failures discussed above). At every 15000 simulation seconds, one node is removed at random. As time goes by, the probability of failure increases (i.e., the probability of the information failing to survive). Once all replicas have disappeared from the system, we count the remaining number of nodes in the system. We ran each algorithm 50 times, and we present the average and deviation in Table III. This information provides the minimum number of nodes below which the system is not viable.

We observe that the repulsion variants need more nodes than their variants without repulsion. Experiments with other shapes have demonstrated that small anchor areas emphasize this difference.

5.2.6 Additional Comments. The Random Way Point model, used for the simulations, is such that the nodes are in the middle of the environment most of the time. So when an anchor area is on the border of the environment, the system will not work well due to a lack of nodes. Alternatively, if the mobility model does not follow the Random Way Point model (e.g., Random Walk), again, the density of the nodes may not be guaranteed in the anchor area, and results will not be the same as those reported here.

Information about the anchor area is stored in a matrix. For example, in the scenario of an anchor area of 1200 x 700 meters, the binary matrix has a size of

1200 x 700 bits—each cell is 1 bit, indicating whether or not the location is inside or outside the area. For the gradient matrix, each cell uses 8 bits, the size is 1200 x 700 bytes.

6. CONCLUSION

This article discusses a viral programming approach for performing persistent storage of data at specific spatial locations on top of mobile computing devices. The main goal is to make the data accessible to a maximum number of users in communication range, without flooding all nodes with the data and without sending a prohibitive number of messages among the nodes.

We investigated four algorithms, Broadcast and Attractor Point and their respective variants with Repulsion (to avoid replicating at nearby locations). Results show that Broadcast converges very quickly, but needs high levels of memory and employs more messages than the Attractor Point. The variants with Repulsion consume less memory and their memory consumption is scalable, but the number of messages among nodes is very high. Attractor Point (without repulsion) does not outperform the other algorithms for all metrics, but reaches high levels of accessibility even for low number of nodes in the area, converges rather quickly in covering the whole area, and employs fewer messages and slightly more memory than its variant with Repulsion.

Future work will concentrate on developing a complete spatial memory concept, including both storage and retrieval of data following a viral programming model. Actual implementations of the storage algorithms on G1 phones are currently being performed.

REFERENCES

ABELSON, H., BEAL, J., AND SUSSMAN, G. J. 2007. Amorphous computing. Tech. rep. MIT-CSAIL-TR-2007-030, Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA.

BEAL, J. 2003. Persistent nodes for reliable memory in geographically local networks. Tech. rep. AI Memo 2003-011, Artificial Intelligence Laboratory, MIT, Cambridge, MA.

BORCEA, C., INTANAGONWIWAT, C., KANG, P., KREMER, U., AND IFTODE, L. 2004. Spatial programming using smart messages: Design and implementation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*.

BUTERA, W. 2007. Text display and graphics control on a paintable computer. In *Proceedings of the 1st International Conference on Self-Adaptive and Self-Organizing Systems (SASO'07)*. IEEE, Los Alamitos, CA, 45–54.

CHENG, J., CHENG, W., AND NAGPAL, R. 2005. Robust and self-repairing formation control for swarms of mobile agents. In *Proceedings of the 20th National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, CA, 59–64.

CORBET, D. J. AND CUTTING, D. 2006. Ad loc: Location-based infrastructure-free annotation. In *Proceedings of the 3rd International Conference on Mobile Computing and Ubiquitous Networking (ICMU'06)*.

COURTS, L., KILLIJIAN, M.-O., POWELL, D., AND ROY, M. 2005. Sauvegarde cooprative entre pairs pour dispositifs mobiles. In *Proceedings of the 2nd French-Speaking Conference on Mobility and Ubiquity Computing (UbiMob'05)*. ACM, New York, 97–104.

DATTA, A., QUARTERONI, S., AND ABERER, K. 2004. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In *Proceedings of the International Conference on Semantics of a Networked World (IC-SNW'04)*. Lecture Notes in Computer Science, vol. 3226, Springer, Berlin, 126–143.

DI MARZO SERUGENDO, G., VILLALBA CASTRO, A., AND KONSTANTAS, D. 2007. Dependable requirements for hovering information. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. Supplemental vol., 36–39.

DOLEV, S., GILBERT, S., LYNCH, N. A., SHVARTSMAN, A. A., AND WELCH, J. 2003. Geoquorums: Implementing atomic memory in mobile ad hoc networks. In *Distributed Computing*, Lecture Notes in Computer Science, vol. 2848, Springer, Berlin, 306–320.

DOLEV, S., GILBERT, S., LYNCH, N. A., SCHILLER, E., SHVARTSMAN, A. A., AND WELCH, J. L. 2004. Virtual mobile nodes for mobile ad hoc networks. In *Distributed Computing*. Lecture Notes in Computer Science, vol. 3274, Springer, Berlin, 230–244.

DOLEV, S., GILBERT, S., LAHIANI, L., LYNCH, N. A., AND NOLTE, T. 2005a. Timed virtual stationary automata for mobile networks. In *Principles of Distributed Systems (OPODIS)*. Lecture Notes in Computer Science, vol. 3974, Springer, Berlin, 130–145.

DOLEV, S., GILBERT, S., SCHILLER, E., SHVARTSMAN, A. A., AND WELCH, J. 2005b. Autonomous virtual mobile nodes. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC'05)*. ACM, New York, 62–69.

EUGSTER, P., FELBER, P., GUERRAOUI, R., AND KERMARREC, A.-M. 2003. The many faces of publish/subscribe. *ACM Comput. Surv. 35*, 2, 114–131.

EUGSTER, P., GARBINATO, G., AND HOLZER, A. 2005. Location-based publish/subscribe. In *Proceedings of the 4th IEEE Symposium on Network Computing and Applications*. IEEE, Los Alamitos, CA.

EUGSTER, P., GARBINATO, G., HOLZER, A., AND LUO, J. 2009. Effective location-based publish/subscribe in manets. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*. IEEE, Los Alamitos, CA.

FEKETE, S. P., SCHMIDT, C., WEGENER, A., AND FISCHER, S. 2006. Hovering data clouds for recognizing traffic jams. In *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (IEEE-ISOLA)*. IEEE, Los Alamitos, CA, 213–218.

FERNANDEZ-MARQUEZ, J. L. AND ARCOS, J. L. 2009. Keeping diversity when exploring dynamic environments. In *Proceedings of the 24th Annual ACM Symposium on Applied Computing*. D. Shin Ed., ACM, New York, 1192–1196.

FERNANDEZ-MARQUEZ, J. L., ARCOS, J. L., AND DI MARZO SERUGENDO, G. 2010. In *Proceedings of the ACM Symposium on Applied Computing (SAC'10)*. ACM, New York.

FERNANDEZ-MARQUEZ, J. L., DI MARZO SERUGENDO, G., AND ARCOS, J. L. 2010. Infrastructureless spatial storage algorithms. Tech. rep. BBKCS-10-05, School of Computer Science and Information Systems, Birkbeck, University of London.

KILLIJIAN, M.-O., POWELL, D., BANTRE, M., COUDERC, P., AND ROUDIER, Y. 2004. Collaborative backup for dependable mobile applications. In *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC'04)*. ACM, New York, 146–149.

LEONTIADIS, I. AND MASCOLO, C. 2007a. Geopps: Opportunistic geographical routing for vehicular networks. In *Proceedings of the IEEE Workshop on Autonomic and Oportunistic Communications* (Colocated with *WOWMOM'07*). IEEE, Los Alamitos, CA.

LEONTIADIS, I. AND MASCOLO, C. 2007b. Opportunistic spatio-temporal dissemination system for vehicular networks. In *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp'07)*. ACM, New York, 39–46.

LEONTIADIS, I., COSTA, P., AND MASCOLO, C. 2009. Persistent content-based information dissemination in hybrid vehicular networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*. IEEE, Los Alamitos, CA.

MAIHFER, C. 2004. A survey of geocast routing protocols. In *IEEE Communications Surveys and Tutorials*, Vol. 6, 32–42.

MAMEI, M. AND ZAMBONELLI, F. 2001. Programming pervasive and mobile computing applications: The TOTA approach. *ACM Trans. Softw. Eng. Method. 2*, 3.

MAMEI, M. AND ZAMBONELLI, F. 2005. Programming stigmergic coordination with the tota middleware. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. ACM, New York, 415–422.

MOTANI, M., SRINIVASAN, V., AND NUGGEHALLI, P. S. 2005. Peoplenet: Engineering a wireless virtual social network. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom'05)*. ACM, New York, 243–257.

ROTH, J. 2003. The critical mass problem of mobile ad-hoc networks. In *Proceedings of the IADIS International Conference e-Society*. IADIS Press, 243–250.

SCELLATO, S., MASCOLO, C., MUSOLESI, M., AND LATORA, V. 2007. Epcast: Controlled dissemination in human-based wireless networks by means of epidemic spreading models. CoRR abs/0711.2780.

VILLALBA CASTRO, A., DI MARZO SERUGENDO, G., AND KONSTANTAS, D. 2008. Hovering information–self-organising information that finds its own storage. In *Proceedings of the International IEEE Conference on Sensor Networks, Ubiquitous and Trustworthy Computing (SUTC'08)*. IEEE, Los Alamitos, CA, 193–200.

VILLALBA CASTRO, A., DI MARZO SERUGENDO, G., AND KONSTANTAS, D. 2009. Hovering information–self-organising information that finds its own storage. In *Autonomic Communications*. Springer, Berlin.

WEGENER, A., SCHILLER, E.M., HELLBRCK, H., FEKETE, S.P., AND FISCHER, S. 2006. Hovering data clouds: A decentralized and self-organizing information system. In *Proceedings of the International Workshop on Self-Organizing Systems*. Lecture Notes in Computer Science, vol. 4124, Springer, Berlin, 243–247.

Received August 2009; revised April 2010; accepted June 2010