Hovering Information – Self-Organising Information that Finds its Own Storage

Alfredo A. Villalba Castro, Giovanna Di Marzo Serugendo, and Dimitri Konstantas

Abstract Hovering information is a mobile computing paradigm where pieces of self-organising information are responsible to find their own storage on top of a dynamic set of mobile devices. Once deployed, the hovering information service acts as a location-based service for disseminating geo-localised information generated by and aimed at mobile users. It supports a wide range of pervasive applications, from urban security to stigmergy-based systems. A piece of hovering information is attached to a geographical point, called the anchor location, and to its vicinity area, called the anchor area. A piece of hovering information is responsible for keeping itself alive, available and accessible to other devices within its anchor area. It does not rely on any central server. This chapter presents the hovering information model and results of simulations performed using replication and caching algorithms involving up to 200 distinct pieces of hovering information in a small geographic area.

1 Introduction

User generated content is taking a large part of the Internet with social networking web sites such as YouTube or MySpace. The equivalent of these sites for mobile users, such as the GyPSii[1] social networking web site, now combine both user-

Alfredo A. Villalba Castro
Centre Universitaire d'Informatique, University of Geneva, Battelle bâtiment A, 7 route de Drize, CH-1227 Carouge, Switzerland, e-mail: alfredo.villalba@unige.ch

Giovanna Di Marzo Serugendo
School of Computer Science and Information Systems, Birkbeck, University of London, Malet Street, London WC1E 7HX, UK, e-mail: dimarzo@dcs.bbk.ac.uk

Dimitri Konstantas
Centre Universitaire d'Informatique, University of Geneva, Battelle bâtiment A, 7 route de Drize, CH-1227 Carouge, Switzerland, e-mail: dimitri.konstantas@unige.ch

[1] www.gypsii.com

generated content and location-based services. Among other, they allow groups of mobile users to share dynamic real-time content or retrieve themselves on a map.

Location-based services usually rely on base stations and from there possibly to the whole Internet to provide some requested information to a mobile user (e.g. what is the nearby Chinese restaurant or where is my friend's car for a user-generated content). This solution has clear advantages such as providing access to large computing capabilities and broadband network access that go beyond those of mobile phones or PDAs.

However, it is not always possible or desired to rely on a central server in particular for user-generated content: extra-terrestrial systems need local communication infrastructures, they cannot communicate with an Earth-based server, or if in a hostile environment cannot rely entirely on a single server; after a natural disaster, when no more infrastructure is available, local communications among available devices help coordination among emergency services; finally, for reliability reasons, it is not always possible to rely on a centralised server representing a single point of failure.

Hovering Information [15] is a concept characterising self-organising information responsible to find its own storage on top of a highly dynamic set of mobile devices. This is a location-aware service for mobile users (people, cars, robots, etc.) that supports dissemination of user-generated geo-localised data among a highly mobile set of devices. This service exploits the mobile devices themselves as a physical support and do not make use of a server. The main requirement of a single piece of hovering information is to keep itself stored in the vicinity of some specified location, which we call the anchor location, 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 anchor location, the information has to hop - "hover" - to another device.

Current services supporting geo-localised data, are deployed using one of the following approaches: centralised servers, virtual structured overlay network offering a stable virtual infrastructure, or direct communication among the mobile nodes themselves. In all these approaches, the mobile nodes decide when and to whom the information is to be sent. Here we take the opposite view; it is the information that decides upon its own storage and dissemination. This opens up other possibilities, not available for traditional MANET services, such as different pieces of hovering information all moving towards the same location and (re-)constructing there a coherent larger information for a user, e.g. TV or video streaming on mobile phones.

A piece of hovering information is a *self-organising* user-defined piece of data which does not need a central server to exist. Individual pieces of hovering information each use local information, such as direction, position, power and storage capabilities of nearby mobile devices, in order to select the next appropriate location. Hovering information benefits from the storage space and communication capacities of the underlying mobile devices.

Main dependability requirements of hovering information are *survivability*, *availability* and *accessibility*. Survivability means that the information is alive somewhere in the environment (i.e. it is stored in some device) but not necessarily close to its anchor location. Availability means that the information has found storage

in the vicinity of the anchor location. Accessibility combines both availability and communication range of wireless mobile devices, and represents the possibility for a user located in the anchor location to access hovering information stored on nearby devices.

The hovering information service requires the following: mobile nodes with computing capacity; direct wireless communications among mobile node such as Bluetooth or WiFi; and a location tracking capability such as a GPS (mobile PDAs), relative distances calculations or light-colour tracking (robots).

This chapter presents the hovering information model as well as replication and caching algorithms allowing multiple pieces of hovering information to get attracted to their respective anchor locations.

Section 2 discusses potential applications of this concept. Section 3 presents the hovering information concept and model. Section 4 discusses replication and caching algorithms, in particular the Attractor Point Algorithm that we have designed where the information is ”attracted” by the anchor location and keeps coming back to this location, and the Location Based Caching algorithm aiming at reducing the number of hovering information stored in the different nodes, when memory is limited. Section 5 reports on simulation results involving up to 200 distinct pieces of hovering information. Finally Section 6 compares our approach to related works, and Section 7 discusses some future works.

2 Applications

This section highlights some future applications in very different areas that could all be developed from the concept of hovering information.

Urban Security The environment considered for this application is a dense urban area where each person carries a GPS enabled device. A hovering information service is available on the device, which allows users to enter comments or warnings related to dangers in the urban environment. Different types of information can be disseminated by the user: warnings about holes in the road or about the existence of thieves (pick-pockets); or comments like ”this corridor is dark and I feel a danger”. Each person entering the area where the information is kept will potentially receive it. Each user has a ”profile” and chooses what types of dangers are relevant to her. For example a blind person will be interested in holes on the road; a weight lifter is not really concerned to be attacked by a thief, while an elderly will find these two pieces of information particularly relevant for her. Similarly, policeman and security guards operating in the same local urban area with high-rate crime could exchange information to each other. Each user attracts different information depending on his profile as soon as it enters the region where the information is located. For such an application the trust/security aspect becomes then crucial, since any GPS owner (including a criminal) may enter fraudulent information. Although we are also working on trust and security

issues related to hovering information, this discussion is beyond the scope of this chapter.

Archaeological Sites The environment in this application is a real archaeological site, most likely an outdoor site, rather large, where visitors just move freely inside it. Users are the visitors of the site. Fixed sensors placed at several locations on the archaeological site provide either current information on actual weather and temperature or historical information about the different locations in the archaeological site. Users visiting the archaeological site wear mobile devices carrying information specific to the user itself (e.g. adult, child, man, woman, teacher, etc) or more likely about the virtual character they want to learn more about. The user receives on his mobile device a virtual reconstructed version of the site as it would be on the day they are visiting: with the same weather conditions, targeted in content to the character they wanted to learn more about, and populated with the other characters that are currently visiting the same location. For instance, consider a group of 3 people (characters) visiting a house in an archaeological site: a cook, a child, and a house's owner. All of them would have a virtual view of how the house looked like at that time. If it is sunny then the view shows a sunny area, if it is cold it could show heating aspects. Each visitor would have a specific tailored explanation (cooking, playing, and owners information) and could visualize the avatars of the others on his mobile device while moving around. There are different types of hovering information going around: visitor-dependent information, weather information, and archaeological/historical information of a specific location in the archaeological site. Personalised information is attracted by the corresponding user's device, aggregates there and shows some virtual view of the site (audio only or both audio/video).

Self-Generative Art Self-generative art [8] refers to art practice where inputs from the creator of the piece of art are assembled together according to some rules (algorithm), such that the resulting piece of art, generated by a computer, is a real-time unfolding work which may display randomness, evolutionary aspects, or self-organising (swarm) behaviour. The piece of art may be music, painting, 3D construction, writing, etc. In this case, the users/creators would then be the multiple visitors of a "learning art experience centre". A piece of art could be a large scale 3D virtual shape produced by inputs provided by each visitor: location in the experience area, weight/height and behaviour (jumping/walking), preferred colour or shape. The virtual shape would have holes were people are currently placed, and bumps where they have left; or heart beating bumps if they are jumping. Rules for combining the different inputs could vary: assembling the virtual surface according to actual or relative distances in the real world, summing up the colours and weights according to different algorithms, keeping visitors input for a random amount of time after they have left the experience area, etc. Sensors are required to determine the weight/size of the person. This value will then have an impact on the final virtual surface. Heavier people or groups of people will provide heavier holes, etc. People moving across the surface create temporary paths across it (that would dissolve completely after a certain time). People carry mobile device through which they provide additional personalised infor-

mation: preferred colour and shape. Each input provided by the different visitors is a piece of hovering information whose goal is to travel to the centre of the experience area and aggregate there with the others in order to provide a visual 3D shape.

Intravehicular Networks Virtual tags are inserted at specific locations on roads or motorways either by cars' drivers or traffic management staff. The purpose here is to provide information to cars' drivers about road conditions, accidents, etc. In such a scenario, using the notion of hovering information, such tags will not be stored on a specific server and made available to users when they reach the zone of interest of the information. Instead the tags are locally stored in the cars and made available through wireless channels to nearby cars. Since data have a meaning for the specific location they have been attributed, data will have to "change" car as soon as the car they are currently stored in leaves the area of the anchor location. The data will then hop from one car to the next one.

Emergency Scenarios In an emergency scenario, virtual data present before a disaster may want to "survive" by using emergency crew or survivors devices. This data can also present useful information for emergency services. Additionally, disaster's survivors may want to indicate their position by placing the appropriate hovering information attaching it to their own location. Emergency crew member can place hovering information to areas where survivors have been found or where there is a chance to find some survivors. In this case, the information will hop from one emergency/survivor device to another one.

Stigmergy Stigmergy is an indirect communication mechanism among individual components of a self-organising system. Communication occurs through modification brought to local environment. The use of ant pheromone is a well known example of stigmergy. Users that communicate by placing hovering information at a geo-referenced position, which is later on retrieved by other users is also an example of stigmergy. The hovering information concept, using an infrastructure free storage media, naturally supports stigmergy-based applications that need to be deployed on an ad hoc manner (e.g. unmanned vehicles or robots). As pointed out by [13] most stigmergy-based deployed systems use a server to support diffusion of artificial pheromone, they do not actually attach the pheromone to physical supports. Hovering information provides a way to store digital pheromone among a group of robots or unmanned vehicles using the robots themselves as physical support for the pheromone. Robots, producing pheromone that needs to be deposited at a certain geographical area, will deposit it under the form of a piece of hovering information. It will then stay located where it has been produced by hovering among the robots present in this area making it available for those robots wishing to retrieve it.

3 Hovering Information Concept

This section formally defines the notion of hovering information system, as well as the three main dependability requirements: survivability, availability and accessibility.

3.1 Coordinates, Distances and Areas

We denote by $\mathbb{E}$ the set of all pairs of geographic coordinates,

$$\mathbb{E} = [-90, 90] \times [-180, 180].$$

A *geographic coordinate a* is a pair:

$$a = (lat, long), \text{ and } a \in \mathbb{E}.$$

North latitude and East longitude are positive coordinates, while South latitude and West longitudes are negative coordinates. We do not consider here depths and heights.

An *area* $A(a,r)$ is defined as the disk whose centre is the geographic coordinate a and has a positive *radius* $r \in \mathbb{R}^+$:

$$A(a,r) = \{b \in \mathbb{E} \mid dist(a,b) < r\}.$$

We consider $dist(a,b)$ to be the distance in meters between two locations on a sphere, provided by any reliable method. See for instance[2].

3.2 Mobile Nodes

Mobile nodes represent the storage and motion media exploited by pieces of hovering information. They are defined as follows. A *mobile node n* is a tuple:

$$n = (id, loc, speed, dir, r_{comm}),$$

where:

[2] http://www.fcc.gov/mb/audio/bickel/distance.html

$id \in \mathscr{I}$ is a mobile node identifier,

$loc \in \mathbb{E}$ is a geographic coordinate location,

$speed \in \mathbb{R}^+$ is a speed in m/s

$dir \in \mathbb{E}$ is a relative geographic coordinate location,

$r_{comm} \in \mathbb{R}^+$ is the communication radius in meters.

We denote by $\mathscr{I}$ the set of all mobile nodes identifiers. When referring to the *id*, *loc* or other field of a mobile node *n*, we will use the following notation $id(n)$, $loc(n)$, etc. Field $loc(n)$ represents the current location of node *n*, while $dir(n)$ is a vector representing the direction of its most recent movement. The range of communication r_{comm} is the maximum distance in meters within which the mobile node may communicate wirelessly with another mobile node.

Let's consider $\mathscr{N}$ a set of mobile nodes, we consider that identifiers of mobile nodes are unique, and we will say that $\mathscr{N}$ is *well defined* if:

$$\forall n_1, n_2 \in \mathscr{N}, (id(n_1) = id(n_2)) \Rightarrow (n_1 = n_2).$$

Given a mobile node *n* with location $loc(n)$ and communication radius $r_{comm}(n)$, the *communication area* of *n*, $A_C(n)$, is the subset of $\mathbb{E}$ given by:

$$A_C(n) = A(loc(n), r_{comm}(n)).$$

Figure 1 shows three mobiles nodes, *m*, *n*, and *p*. While the communication range of *m* is enough to let it be in range of both *n* and *p*, the communication range of *n* and *p* being much smaller prevents them to be directly in range.

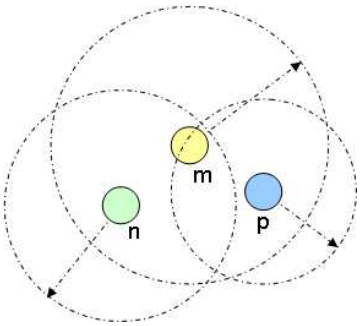

Fig. 1 Mobile Nodes and Communication Range

3.3 Hovering Information

Let $\mathscr{N}$ be a well defined set of mobile nodes. A *piece of hovering information* h is a tuple:

$$h = (id, a, r, n, data, policies, size),$$

where:

$$\begin{aligned}
id &\in \mathscr{I} \text{ is a hovering information identifier,}\\
a &\in \mathbb{E} \text{ is the anchor location,}\\
r &\in \mathbb{R}^+ \text{ is the anchor radius,}\\
n &\in \mathscr{N} \text{ is the mobile node where } h \text{ is currently located,}\\
data &\text{ is the data carried by } h,\\
policies &\text{ are the hovering policies of } h,\\
size &\in \mathbb{N}^+ \text{ is the size of } h \text{ in bytes.}
\end{aligned}$$

We denote by $\mathscr{I}$ the set of all hovering information identifiers. When referring to the id, a or other field of a hovering information h, we will use the following notation $id(h)$, $a(h)$, etc. Policies stand for hovering policies stating how and when a piece of hovering information has to hover. The size is an important element of a single piece of hovering information; however the simulation algorithms presented in this chapter are not yet using this notion.

A piece of hovering information h is a piece of data whose main goal is to remain stored in an area centred at a specific location called the *anchor location* $a(h)$, and having a radius $r(h)$, called the *anchor radius*.

The *anchor area* of h, $A_H(h)$, is the disk whose centre is the anchor location $a(h)$ and whose radius is $r(h)$:

$$A_H(h) = A(a(h), r(h)).$$

Let $\mathscr{H}$ be a set of pieces of hovering information. We consider that identifiers of pieces of hovering information are unique, but replicas (carrying same data and anchor information) are allowed on different mobile nodes, and we will say that $\mathscr{H}$ is *well defined* iff:

$$\begin{aligned}
\forall h_1, h_2 \in \mathscr{H}, (h_1 \neq h_2) \Rightarrow\\
&(id(h_1) \neq id(h_2)) \vee\\
&((id(h_1) = id(h_2)) \wedge (a(h_1) = a(h_2)) \wedge\\
&(r(h_1) = r(h_2)) \wedge (data(h_1) = data(h_2)) \wedge\\
&(n(h_1) \neq n(h_2))).
\end{aligned}$$

Let $\mathscr{H}$ be a well defined set of pieces of hovering information. Let $h \in \mathscr{H}$ be a hovering information, a *replica* h_r of h is a piece of hovering information $h_r \in \mathscr{H}$ such that:

$$id(h) = id(h_r) \wedge n(h) \neq n(h_r).$$

From now on, we will consider only well defined sets $\mathscr{H}$ of pieces of hovering information, where pieces of hovering information with the same *id* are either the same or a replica of each other. We also consider that there is only one instance of a hovering information in a given node *n*, any other replica resides in another node.

Figure 2 shows a piece of hovering information (blue hexagon) and two mobile nodes (yellow circles). One of them hosts the hovering information whose anchor location, radius and area are also represented (blue circle). The communication range of the second mobile node is also showed.

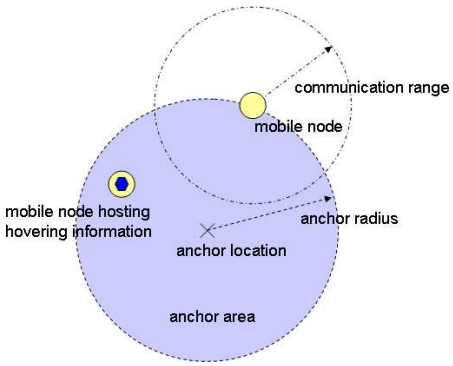

Fig. 2 Mobile Nodes and Hovering Information

Definition 1 (Hovering Information System at time t). A hovering information system at time t, $HoverInfo_t$, is a tuple:

$$HoverInfo_t = (\mathscr{N}_t, \mathscr{H}_t),$$

where $\mathscr{N}_t$ is a well defined set of mobile nodes, $\mathscr{H}_t$ is a well defined set of hovering information over $\mathscr{N}_t$:

$$\forall h \in \mathscr{H}_t \Rightarrow n(h) \in \mathscr{N}_t.$$

A hovering information system at time t is a snapshot (at time t) of the status of the system. Mobile nodes can change location, new mobile nodes can join the system, others can leave. New pieces of hovering information can appear (with new identifiers), replicas may appear or disappear (same identifiers but located on other nodes), hovering information may disappear or change node.

Figure 3 shows two different pieces of hovering information h_1 (blue) and h_2 (green), having each a different anchor location and area. Three replicas of h_1 are currently located in the anchor area (nodes n_2, n_3 and n_4), while two replicas of h_2

are present in the anchor area of h_2 (nodes n_2 and n_5). It may happen that a mobile device hosts replicas of different pieces of hovering information, as it is the case in the figure for the mobile node n_2 that is at the intersection of the two anchor areas. The arrows here also represent the communication range possibilities among the nodes.

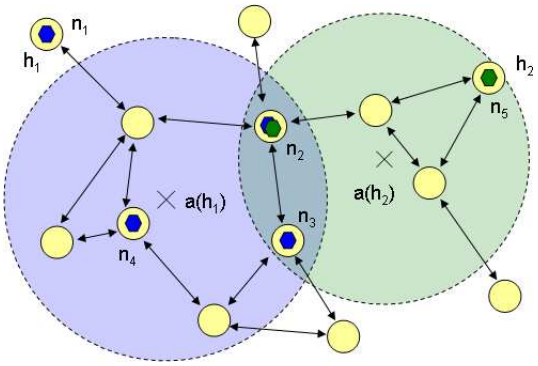

Fig. 3 Hovering Information System at time t

3.4 Notations

Before defining the notions of survivability, availability and accessibility, we will define the following additional notations.

Let's consider $HoverInfo_t = (\mathcal{N}_t, \mathcal{H}_t)$, a Hovering Information System at time t, let h be a piece of hovering information, and $n \in \mathcal{N}_t$ be a mobile node at time t, we denote:

$$
\begin{aligned}
R_H(h,t) &= \{k \in \mathcal{H}_t \mid id(h) = id(k)\},\\
&\quad \text{the set of replicas of a piece of hovering information } h \text{ at time } t;\\
R_N(n,t) &= \{h \in \mathcal{H}_t \mid n(h) = n\},\\
&\quad \text{the set of pieces of hovering information in node } n \text{ at time } t;\\
P_N(n,t) &= loc(n), \text{the position of node } n \text{ at time } t;\\
N_N(n,t) &= \{m \in \mathcal{N}_t \mid (dist(loc(m), loc(n)) < r_{comm}(n)) \vee\\
&\quad (dist(loc(m), loc(n)) < r_{comm}(m)),\}\\
&\quad \text{the set of neighbouring nodes of node } n \text{ at time } t;\\
S(X) &= \text{the surface area of a surface } X.
\end{aligned}
$$

Since $\mathscr{H}_t$ is well defined, there are no two different pieces of hovering information (with different data) referred by the same identifier. It is also important to notice that h does not necessarily belong to $\mathscr{H}_t$, it may have disappeared from the system, but some of its replicas are still located in some mobile nodes. We consider that $R_N(n,t)$ is actually a set (not a multi-set), i.e. there are no copies of the same hovering information stored at the same location. The neighbouring nodes in $N_N(n,t)$ are those in range of communication to n.

3.5 Properties - Requirements

3.5.1 Survivability

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

Definition 2 (Survivability of Hovering Information h at time t). Let $HoverInfo_t = (\mathscr{N}_t, \mathscr{H}_t)$ be a Hovering Information System at time t. Let h be a piece of hovering information, the survivability of h at time t is given by the boolean value:

$$sv_H(h,t) = \begin{cases} 1 & \text{if } \exists n \in \mathscr{N}_t, R_H(h,t) \cap R_N(n,t) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

The 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 3 (Rate of Survivability of Hovering Information h at time t). Let h be a piece of hovering information, the survivability of h between time t_c (creation time of h) and time t is given by:

$$SV_H(h,t) = \frac{1}{t-t_c} \sum_{\tau=t_c}^{t} sv_H(h,\tau).$$

3.5.2 Availability

A hovering information is available at some time t if there is at least a node in its anchor area hosting a replica of this information.

Definition 4 (Availability of Hovering Information h at time t). Let $HoverInfo_t = (\mathscr{N}_t, \mathscr{H}_t)$ be a Hovering Information System at time t. Let h be a piece of hovering information, the availability of h at time t is given by:

$$av_H(h,t) = \begin{cases} 1 & \text{if } \exists n \in \mathscr{N}_t, (P_N(n,t) \in A_H(h)) \wedge (R_H(h,t) \cap R_N(n,t) \neq \emptyset) \\ 0 & \text{otherwise.} \end{cases}$$

The availability of a piece of hovering information along a period of time is defined as the rate between the amount of time along which this information has been available during this period and the overall time.

Definition 5 (Rate of Availability of Hovering Information h at time t). Let h be a piece of hovering information, the availability of h between time t_c (creation time of h) and time t is given by:

$$AV_H(h,t) = \frac{1}{t-t_c} \sum_{\tau=t_c}^{t} av_H(h,\tau).$$

3.5.3 Accessibility

We distinguish availability from accessibility in the following way: a piece of hovering information (or one of its replica) present on some node located in the anchor area is said to be available. However, such a piece of hovering information may not be accessible to a mobile which is far apart from the mobile node where the hovering information (or its replica) is actually stored.

A hovering information is accessible by a node n at some time t if the node is able to get this information. In other words, if it exists a node m being in the communication range of the interested node n and which contains a replica of the piece of hovering information.

Definition 6 (Accessibility of Hovering Information h for node n at time t). Let $HoverInfo_t = (\mathscr{N}_t, \mathscr{H}_t)$ be a Hovering Information System at time t. Let h be a piece of hovering information, let $n \in \mathscr{N}_t$ be a mobile node, the accessibility of h for n at time t is given by:

$$ac_H(h,n,t) = \begin{cases} 1 & \text{if } \exists m \in \mathscr{N}_t, (m \in N_N(n,t)) \wedge (R_H(h,t) \cap R_N(m,t) \neq \emptyset) \\ 0 & \text{otherwise.} \end{cases}$$

We also define the accessibility of a piece of hovering information as the rate between the covered area by the hovering information's replicas and its anchor area.

Definition 7 (Accessibility of Hovering Information h at time t). Let $HoverInfo_t = (\mathscr{N}_t, \mathscr{H}_t)$ be a Hovering Information System at time t. Let h be a piece of hovering information, the accessibility of h at time t is given by:

$$ac_H(h,t) = \frac{S(\bigcup_{r \in R_H(h,t)} A_C(n(r)) \cap A_H(h))}{S(A_H(h))},$$

where $S(X)$ denotes the surface of X.

The accessibility along a period of time is defined as the average of the accessibility through that period of time.

Definition 8 (Rate of Accessibility of Hovering Information h at time t). Let h be a piece of hovering information, the accessibility of h between time t_c (creation time of h) and time t is given by:

$$AC_H(h,t) = \frac{1}{t-t_c} \sum_{\tau=t_c}^{t} ac_H(h,\tau).$$

Let us notice that an available piece of hovering information is not necessarily accessible and vice-versa, an accessible piece of hovering information is not necessary available. Figure 4 shows different cases of survivability, availability and accessibility. In Figure 4(a), hovering information h (blue) is not available, since it is not physically present in the anchor area, however it is survival as there is a node hosting it. In Figure 4(b), hovering information h is now available as it is within its anchor area, however it is not accessible from node n_1 because of the scope of the communication range. Finally, in Figure 4(c), hovering information h is survival, available and accessible from node n_1.

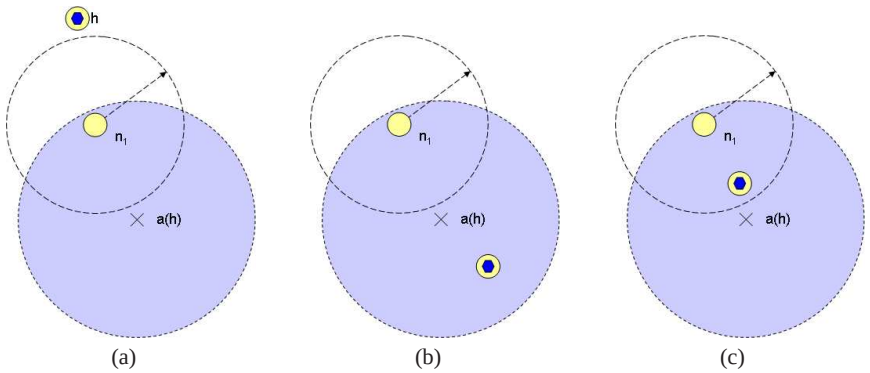

Fig. 4 Survivability, Availability and Accessibility

It is thus important to distinguish availability from accessibility: a piece of hovering information may be available (i.e. present) in the anchor area, but due to actual communication ranges among the nodes, it is not necessarily accessible for all nodes into the anchor area.

Similarly, it is interesting to note that in some situations a piece of hovering information even though not available at its anchor location could be accessible for some nodes provided there is a node in communications range hosting h.

4 Algorithms for Hovering Information

Survivability, availability, and accessibility are among the most fundamental issues of hovering information as we discussed in [15] and [3]. Security and trust issues are important issues when considering hovering information, however they go beyond the scope of this chapter, and will not be discussed here. Survivability addresses the problem of keeping a piece of hovering information alive as long as defined by the information itself. Availability deals with the problem of keeping the information present in its anchor area while accessibility relates to the possibility for a user to access a piece of hovering information stored on a device which is in communication range.

As mentioned in the previous section, these notions are closely related to each other, but none of them necessarily implies the others.

This chapter focuses on the study of the survivability and availability of pieces of hovering information. We propose an *Attractor Point* algorithm, whose aim is to keep the hovering information alive and available in its anchor area as long as possible. An anchor location a acts as an attractor point: all pieces of hovering information that have a as anchor location tend to converge towards a.

Besides the Attractor Point algorithm, we describe a *Broadcast* algorithm which is expected to have better survivability and availability performances than the Attractor Point, but at the cost of being more memory and network greedy. We use the Broadcast algorithm as a comparison threshold. Pieces of hovering information periodically broadcast (replicate) themselves to all the nodes in the communication range.

Mobile nodes have a limited memory and so cannot store an infinite number of hovering information replicas. We study two different caching policies: *Location-Based Caching* and *Generation-Based Caching*. The Location-Based Caching policy decides whether to remove or keep a replica on the basis of the current position of the node (or the replica), its proximity to the anchor location, and the portion of the anchor area covered by the communication area of the node. The Generation-Based Caching policy takes the decision of removing a replica based on the generation of the replica, removing those replicas that have been replicated most.

4.1 Assumptions

We make the following assumptions in order to keep the problem simple while focusing on measuring availability and resource consumption.

Limited memory All mobile nodes have a limited amount of memory able to store hovering information replicas. The proposed algorithms take into account the remaining memory space.

Uniform size All pieces of hovering information have the same size and the caching algorithms do not take in consideration the size as a criteria when removing a replica.

Unlimited energy All mobile nodes have an unlimited amount of energy. The proposed algorithms do not consider failure of nodes or impossibility of sending messages because of low level of energy.

Instantaneous processing Processing time of the algorithms in a mobile node is zero. We do not consider performance problems related to overloaded processors or execution time.

In-built geo-localization service Mobile nodes have an in-built geo-localization service such as GPS which provides the current position. We assume that this information is available to pieces of hovering information.

Velocity vector service Mobile nodes have an in-built velocity vector service providing the instantaneous speed and direction of the node. We assume that this information is available to pieces of hovering information.

Neighbours discovering service Mobile nodes are able to get a list of their current neighbouring nodes at any time. This list contains the position, speed, and direction of the nodes. As for the other two services, this information is available to pieces of hovering information.

4.2 Safe, Risk and Relevant Areas

Hovering policies are attached to pieces of hovering information. We consider here that all pieces of hovering information have the same hovering policies: active replication and hovering in order to stay in the anchor area (for availability and accessibility reasons), hovering and caching when too far from the anchor area (survivability), and cleaning when too far from the anchor area to be meaningful (i.e. disappearance). The decision on whether to replicate itself or to hover depends on the current position of the mobile device in which the hovering information is currently stored.

Given an anchor area $A(a,r)$, the *safe area* $A(a,r_{safe})$ is the disk whose centre is the anchor location a and whose radius is the *safe radius* r_{safe}, a positive radius smaller than the anchor radius r, i.e. $r_{safe} < r$ and $r_{safe} \in \mathbb{R}^+$:

$$A(a,r_{safe}) = \{b \in \mathbb{E} \mid dist(a,b) < r_{safe}\}.$$

A piece of hovering information located in the safe area can safely stay in the current mobile node, provided the conditions on the node permit this: power, memory, etc.

Given an anchor area $A(a,r)$, a risk area is a ring centred at the anchor location, which overlaps with the anchor area and is limited by the safe area.

The *risk area* $R(a,r_{safe},r_{risk})$ is the ring given by;

$$R(a, r_{safe}, r_{risk}) = A(a, r_{risk}) \backslash A(a, r_{safe}),$$

where $r_{safe} < r < r_{risk}$ and $r_{risk}, r_{safe} \in \mathbb{R}^+$.

A piece of hovering information located in the risk area should actively seek a new location on a mobile node going into the direction of the safe area. It is in this area that the hovering information actively replicates itself in order to stay available and in the vicinity of the anchor location.

The *relevant area* limits the scope of survivability of a piece of hovering information. The relevant area $A(a, r_{rel})$ is the disk whose centre is the anchor location a and whose radius is the relevant radius r_{rel} bigger than the risk radius:

$$A(a, r_{rel}) = \{b \in \mathbb{E} \mid dist(a, b) < r_{rel}\},$$

where $r_{risk} < r_{rel}$, and $r_{rel} \in \mathbb{R}^+$.

The ring area $A(a, r_{rel}) \backslash A(a, r_{risk})$ represents the area where the hovering information seeks to survive but does not actively replicate itself (in order to avoid flooding). It may come back to the anchor area through mobile devices going in the direction of the anchor area.

The *irrelevant area* is all the area $U(a, r)$, outside the relevant area, it is given by:

$$U(a, r_{rel}) = \mathbb{E} \backslash A(a, r_{rel}).$$

A piece of hovering information located in the irrelevant area can disappear; it is relieved from survivability goals.

Figure 5 depicts the different types of radii and areas discussed above centred at a specific anchor location a. The smallest disk represents the safe area, the blue area is the anchor area, the ring limited by the risk radius and the safe radius is the risk area, and finally the larger disk is the relevant area.

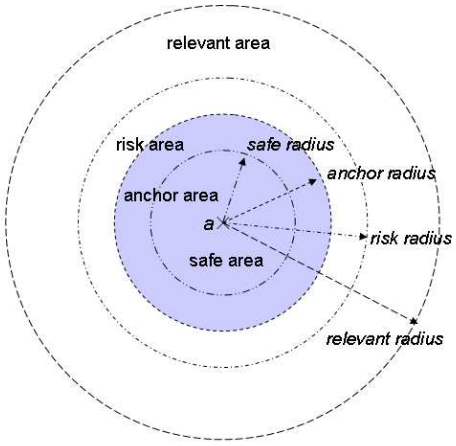

Fig. 5 Radii and Areas

The values of these different radii are different for each piece of hovering information and are typically stored in the Policies field of the hovering information. In the following algorithms we consider that all pieces of hovering information have the same relevant, risk and safe radius.

4.3 Replication

A piece of hovering information h has to replicate itself onto other nodes in order to stay alive, available and accessible. We describe two such replication algorithms for simulating two variants of these policies: the Attractor Point algorithm (AP) and the Broadcast-Based algorithm (BB). Both algorithms are triggered periodically each T_R (replication time) seconds and only replicas of h being in the risk area are replicated onto some neighbouring nodes (nodes in communication range) which are selected according to the replication algorithm.

4.3.1 Attractor Point Algorithm

The anchor location of a piece of hovering information acts constantly as an attractor point to that piece of hovering information and to all its replicas. Replicas tend to stay as close as possible to their anchor area by jumping from one mobile node to the other.

Algorithm 1 Attractor Point Replication Algorithm

```
 1: procedure REPLICATION
 2:     pos ← NODE-POSITION
 3:     N ← NODE-NEIGHBOURS
 4:     P ← NEIGHBOURS-POSITION(N)
 5:     for all repl ∈ REPLICAS do
 6:         a ← ANCHOR-LOCATION(repl)
 7:         dist ← DISTANCE(pos, a)
 8:         if (r_safe ≤ dist ≤ r_risk) then
 9:             D ← DISTANCE(P, a)
10:             M ← SELECT-KR-CLOSESTS(N, D, k_R)
11:             MULTICAST(repl, M)
12:         end if
13:     end for
14: end procedure
```

Periodically and for each mobile node (see Algorithm 1), the position of the mobile node (line 2) is retrieved together with the list and position of all mobile nodes in communication range (lines 3 and 4). Hovering information replicas verify whether they are in the risk area and need to be replicated (line 8). The number of target nodes composing the multicast group is defined by the constant k_R (replication

factor). The distance between each mobile node in range and the anchor location is computed (line 9). The k_R mobile nodes with the shortest distance are chosen as the target nodes for the multicast (lines 10). A piece of hovering information in the risk area multicasts itself to the k_R mobile nodes that are in communication range and closest to its anchor location (line 11).

Figure 6 illustrates the behaviour of the Attractor Point algorithm. Consider a piece of hovering information h in the risk area. It replicates itself onto the nodes in communication range that are the closest to its anchor location. For a replication factor $k_R = 2$, nodes n_2 and n_3 receive a replica, while all the other nodes in range do not receive any replica.

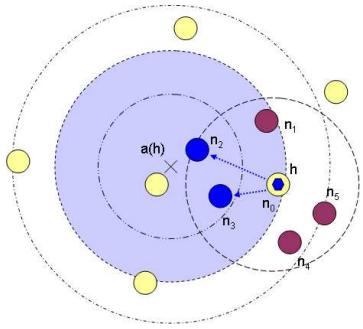

Fig. 6 Attractor Point Algorithm

4.3.2 Broadcast-Based Algorithm

The Broadcast-based algorithm (see Algorithm 2) is triggered periodically (each T_R) for each mobile node. After checking the position of the mobile node (line 2); pieces of hovering information located in the risk area (line 6) are replicated and broadcasted onto all the nodes in communication range (line 7). We expect this algorithm to have the best performance in terms of availability but the worst in terms of network and memory resource consumption.

Figure 7 illustrates the behaviour of the Broadcast algorithm. Consider the piece of hovering information h in the risk area, it replicates itself onto all the nodes in communication range, nodes n_1 to n_5 (blue nodes).

4.4 Caching

In this chapter we assume that nodes have a limited amount of memory to store the pieces of hovering information (replicas). As the number of distinct hovering information increases, so will be the total number of replicas. The buffer of nodes

Algorithm 2 Broadcast-Based Replication Algorithm

```
1: procedure REPLICATION
2:     pos ← NODE-POSITION
3:     for all repl ∈ REPLICAS do
4:         a ← ANCHOR-LOCATION(repl)
5:         dist ← DISTANCE(pos, a)
6:         if (r_safe ≤ dist ≤ r_risk) then
7:             BROADCAST(repl)
8:         end if
9:     end for
10: end procedure
```

will get full at some point and some replicas should have to be removed in order to store new ones.

We present two different caching policies. The first one, called the Location-Based Caching (LBC), decides whether to remove or keep a replica based on the current position of the node (or the replica), its proximity to the anchor location, and the portion of the anchor area covered by the communication area of the node. The second one, called the Generation-Based Caching (GBC), is based on the generation of replicas, the more a replica is old, the more it will have a tendency to disappear as the priority is given to younger replicas.

We compare these caching techniques with a simpler one which only ignores the incoming replicas as soon as there is no free space in the mobile device buffer.

Besides these caching algorithms, it is important to notice that we only consider the position and the generation of replicas. We do not take into consideration caching policies such as the priority, the time-to-live or the replicas size (since all replicas considered in this paper have the same size).

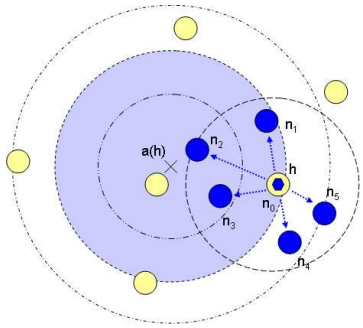

Fig. 7 Broadcast-Based Algorithm

4.4.1 Location-Based Caching

At each node, this caching policy decides to remove a previously stored replica from the node's full buffer, and to replace it by the new incoming replica based on their respective location relevance value. We define the location relevance value of a replica, being this replica already stored in the node's buffer or being a new incoming replica, to its anchor location and area as it follows:

$$relevance = \alpha * area + \beta * proximity,$$

where *area* is the normalised estimation of the overlapping area of the nodes' communication range area and the replica's anchor area, *proximity* is the normalised proximity value between the current position of the node and the anchor location of the replica, α and β are real coefficients having values between 0 and 1 and $\alpha + \beta = 1$.

Each time a new incoming replica arrives (see Algorithm 3), the least location relevant replica is chosen from all the replicas stored in buffer of the node (lines 2 to 10). The location relevance of the incoming replica is computed and compared to that of the least location relevant replica, whatever the original hovering information they refer to (lines 11 and 12). The least location relevant replicas is removed from the buffer and replaced by the incoming replica if the latter has a greater location relevance value (lines 12 to 14). Otherwise, the incoming replica is just discarded (line 16). In this way, the location-based caching algorithm will tend to remove replicas being too far from their anchor location or being hosted in a node covering only a small part of their anchor area.

The location relevance function (see Algorithm 4) computes the location relevance of a replica hosted in a node sitting in the anchor area of the replica. The distance between the location of the node and the anchor location of the replica are computed (lines 2 to 5). The overlapping area of the node's communication range area and the replica's anchor area is estimated (lines 6 to 13). Based on these two values, distance and estimated overlapping area, a normalised overlapping area and a normalised proximity values are computed (lines 14 and 15). Finally, the location relevance of the replica hosted in the node is computed using the previous formula with $\alpha = 0.8$ and $\beta = 0.2$ (lines 16 and 17). Figure 8 illustrates the notion of location relevance.

4.4.2 Generation-Based Caching

We define the generation of a replica in the following way: the first replica created (normally by the user or user application) of a piece of hovering information has a generation 0, when this replica replicates itself then it creates new replicas having generation 1, and so on. The generation of a replica gives us an idea of the number of replicas existing as the process of replication follows an exponential growth. The generation-based caching algorithm tends to remove replicas having a high gener-

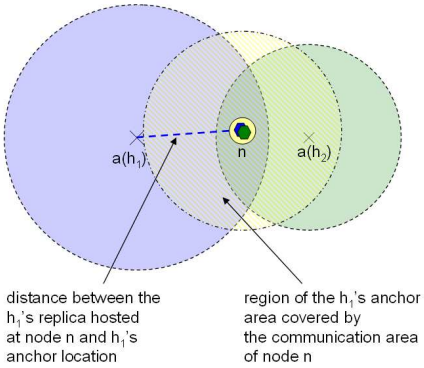

Fig. 8 Location-Based Caching Policy

ation as there are likely more replicas leaving around than a replica having a lower generation.

Each time a new incoming replica arrives (see Algorithm 5), the oldest replica (the one having the highest generation value) is chosen from all the replicas stored in the buffer of the node (lines 2 to 10). The generation of the incoming replica is retrieved and compared to that of the oldest replica, whatever the original hovering information they refer to (lines 11 and 12). The oldest replica is removed from the buffer and replaced by the incoming replica if the latter has a smaller generation value (lines 12 to 14). Otherwise, the incoming replica is just discarded (line 16).

Algorithm 3 Location-Based Caching (LBC)

```
 1: procedure LBC(replica)
 2:     repl_min ← NULL
 3:     rele_min ← MAX_RELEVANCE_VALUE
 4:     for all repl ∈ REPLICAS do
 5:         rele ← RELEVANCE(repl)
 6:         if (rele ≤ rele_min) then
 7:             repl_min ← repl
 8:             rele_min ← rele
 9:         end if
10:     end for
11:     rele ← RELEVANCE(replica)
12:     if (rele > rele_min) then
13:         REMOVE(repl_min, REPLICAS)
14:         INSERT(replica, REPLICAS)
15:     else
16:         DISCARD(replica)
17:     end if
18: end procedure
```

Algorithm 4 Location Relevance Function

```
1: function RELEVANCE(replica)
2:     a ← ANCHOR-LOCATION(replica)
3:     r ← ANCHOR-RADIUS(replica)
4:     pos ← NODE-POSITION
5:     dist ← DISTANCE(pos, a)
6:     area ← 0
7:     if (dist < (r_comm + r)) then
8:         if (dist > (r_comm − r)) then
9:             area ← ((r_comm + r) − dist)^2
10:        else
11:            area ← (2 ∗ r)^2
12:        end if
13:    end if
14:    area ← area/(4 ∗ r^2)
15:    proximity ← (e^(dist/100))^−1
16:    relevance ← 0.8 ∗ area + 0.2 ∗ proximity
17:    return relevance
18: end function
```

4.5 Cleaning

The cleaning algorithm periodically - each T_C (cleaning time) seconds - and for each node, removes the replicas that are too far from their anchor location, i.e. those replicas that are in the irrelevant area. This represents the cases where the replica considers itself too far from the anchor area and not able to come back anymore. This avoids as well the situation were all nodes have a replica.

Algorithm 5 Generation-Based Caching (GBC)

```
1: procedure GBC(replica)
2:     repl_max ← NULL
3:     gen_max ← MIN_GENERATION_VALUE
4:     for all repl ∈ REPLICAS do
5:         gen ← GENERATION(repl)
6:         if (gen ≥ gen_max) then
7:             repl_max ← repl
8:             gen_max ← gen
9:         end if
10:    end for
11:    gen ← GENERATION(replica)
12:    if (gen ≤ gen_max) then
13:        REMOVE(repl_max, REPLICAS)
14:        INSERT(replica, REPLICAS)
15:    else
16:        DISCARD(replica)
17:    end if
18: end procedure
```

5 Evaluation

We evaluated the behaviour of the above described replication algorithms and caching policies under different scenarios by varying the number of nodes and the number of hovering informations.

For a single piece of hovering information, results reported in [16] show that the Attractor Point algorithm reaches availability levels of 80% when the number of nodes in the environment reaches 100 and 93% with 200 nodes. It is thus competitive to the Broadcast-Based algorithm while using much less memory and network traffic.

This Section reports results for multiple distinct pieces of hovering information co-existing at the same time. We measured the average survivability (Definition 3) and availability (Definition 5) as well as another performance metrics (cf. 5.2) such as the messages complexity, replication complexity, overflows, erased replicas and concentration .

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 range of 121m.

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. In this kind of mobility model, a node moves along a straight line with speed and direction changing randomly at some random time intervals.

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_{riks}) of 70m, a relevance 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 having a capacity to 20 different replicas. The caching algorithm is constantly listening for the arrival of new replicas. Table 1 summarises these values.

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; and varying number of different pieces of hovering information existing in the system: from 20 to 200 hoverinfos, increasing the number of pieces by 20. Each of this scenarios has been investigated with different replication algorithms and caching policies.

We have performed 20 runs for each of the above scenarios. One run lasts 3'600 simulated seconds. All the results presented here are the average of the 20 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).

Blackboard	500mx500m
Mobility Model	Random Way Point
Nodes speed	1m/s to 10 m/s
Communication range (r_{comm})	121m
Buffer size	20 replicas
Replication time (T_R)	10s
Cleaning time (T_C)	60s
Replication factor (k_R)	4
Anchor radius (r)	50m
Safe radius (r_{safe})	30m
Risk radius (r_{risk})	70m
Relevant radius (r_{rele})	200m

Table 1 Simulation Settings

5.2 Metrics

In addition to the survivability and availability properties of a hovering information system, we also measured the performance metrics described below.

Messages Complexity. The messages complexity of a hovering information system at a given time t is defined as the total number of messages exchanged by nodes since the initial time. This metric provides as well a feasibility criterion and will serve as a basis to extrapolate actual implementation results, such as latency and overhead of a real system.

Replication Complexity. The replication complexity measures, for a given piece of hovering information h, the maximum number of replicas having existed in the whole system at the same time. In a real implementation this parameter will play an important role since we will prefer algorithms minimizing the replication complexity and maximizing the availability and survivability of hovering information.

Overflows. The overflows of a caching policy stands for the number of times that new incoming replicas have not found enough storage space in a node to be hosted. After an overflow happens, it is up to the caching policies to replace or not an existing stored replica by the new incoming one. We expect that the Location-Based Caching policy will generate less overflows than the other caching polcies.

Erased Replicas. The erased replicas of a caching policy represents the number of times that a node has had to remove a replica from its buffer to store a new incoming replica. The node takes this decision based on the caching policy. We also expect that the Location-Based Caching policy will erase less replicas than the other caching policies.

Concentration. The concentration of a given piece of hovering information *h* is defined as the rate between the number of replicas of *h* present in the anchor area and the total number of replicas of this hovering information in the whole environment. This metric shows how replicas are distributed around a geographical area.

5.3 Results

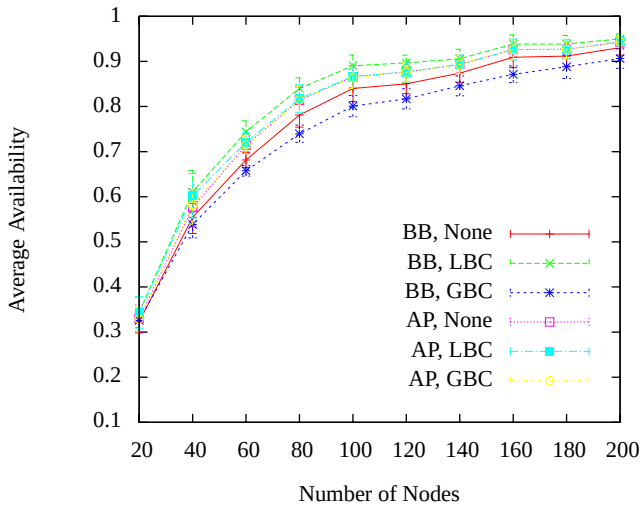

Fig. 9 Availability - 40 Hoverinfos

Figures 9, 10 and 11 show the average availability for the two replication algorithms: Broadcast-Based (BB) and Attractor Point (AP). Each algorithm uses three different caching policies: no caching (None), Location-Based Caching (LBC) and Generation-Based Caching (GBC). Each figure corresponds to a system containing 40, 120 and 200 pieces of hovering information. We observe that BB gets worse results than AP as the number of pieces of hovering information increases. Indeed, the BB tends to overload the system with an exponential growing number of replicas. As each node has a limited buffer size, the latter tend to get full and not all replicas can be accommodated within the buffer size and a large portion of them is discarded. On the other hand, AP manages to better administrate the buffer size producing less replicas which are stored in nodes closer to the anchor location of the replicas. The combination of the AP and LBC keep the information in its anchoring area in a much more optimal way than the other cases.

We also observe that algorithms using the LBC caching policy keep good levels of availability despite the number of nodes or the number of pieces of hovering

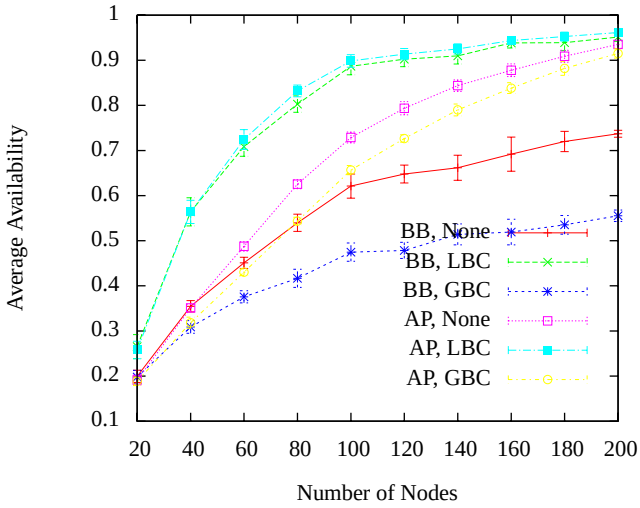

Fig. 10 Availability - 120 Hoverinfos

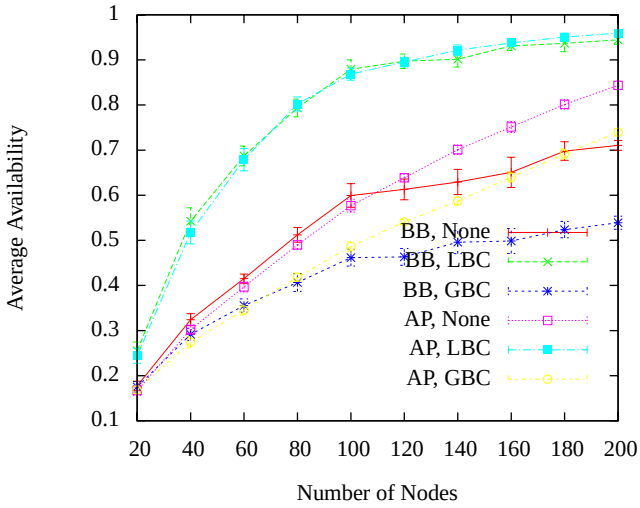

Fig. 11 Availability - 200 Hoverinfos

information, whereas it gets gradually worse for algorithms using the GBC or no caching policy.

Figure 12 depicts the average availability of the AP replication algorithm using the LBC caching policy for different numbers of nodes. For a number of nodes above 120, we notice that the availability is high enough (above 85%) and it keeps quite stable as the number of pieces of hovering information increases. We confirm from

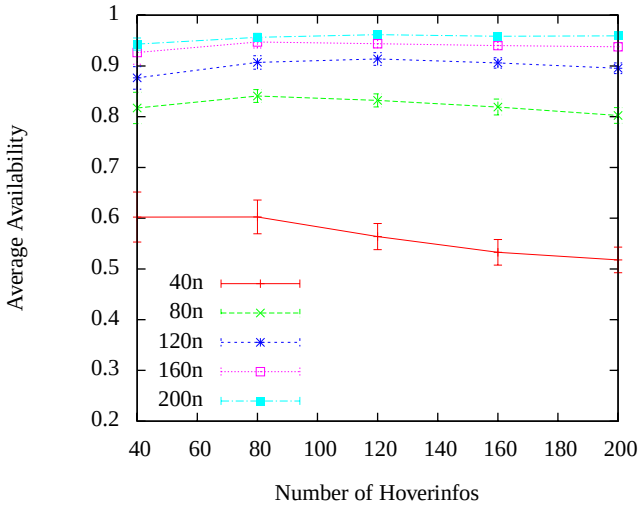

Fig. 12 Availability - Attractor Point with Location-Based Caching

this that the AP with LBC algorithms are scalable in terms of absorption of hovering information (number of distinct pieces of hovering information), since during the experiments with 120 nodes and more, up to 200 distinct hovering information pieces have been accommodated into the system with an availability above 85%. In the case when the number of nodes is 40, we can observe that the absorption limit, for this configuration, has been reached as the availability starts decreasing after 80 pieces of hovering information.

Figure 13 compares the survivability and availability for the AP algorithm in a system composed of 200 pieces of hovering information. As expected, the survivability is higher than the availability in all the cases. This proves the fact that an available piece of hovering information is survival but a survival one is not necessary available. We can also notice that these two metrics have the same shape (for the same algorithm) meaning that they are strongly related and consequently a piece of hovering information with a lot of chances to survive will have a lot of chances to keep itself available as well.

Figure 14 depicts the average number of overflows for the BB and AP using the three different caching policies: None, LBC and GC. We can observe that the BB produces around 10 times more overflows than the AP because of its exponential replication nature. We also see that the number of overflows for the AP tends to stabilize as the number of nodes grows. This is due to the controlled behaviour of the AP algorithm that prevents exponential growth.

Figure 15 shows the average number of replicas erased from the nodes buffer of nodes in order to store new incoming replicas. We observe that the AP algorithm erases much less replicas and in a more stabilized way than the BB algorithm. We also notice that the BB with GBC tends to erase replicas in an exponential way

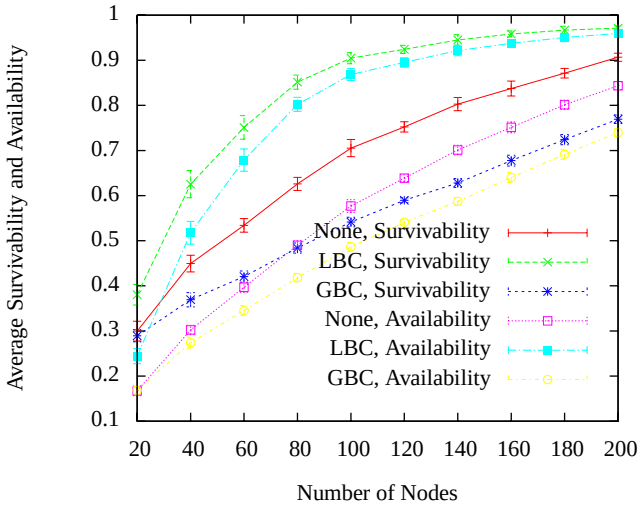

Fig. 13 Survavibility vs Availability - Attractor Point - 200 Hoverinfos

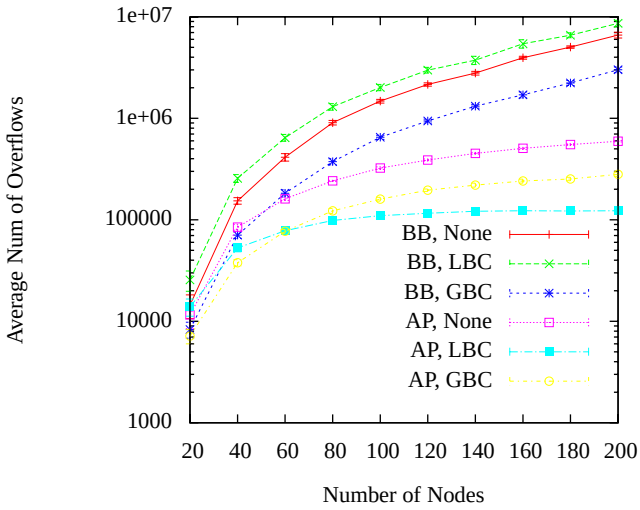

Fig. 14 Overflows - 200 Hoverinfos

which means that the generation-based caching policy combined with the exponential replication behaviour of BB is not a good differentiation factor for caching replicas since this combination of algorithsm tends to insert and erase replicas permanently.

Figure 16 depicts the average number of messages sent by the different algorithms using the different caching policies. We notice that the algorithms using the

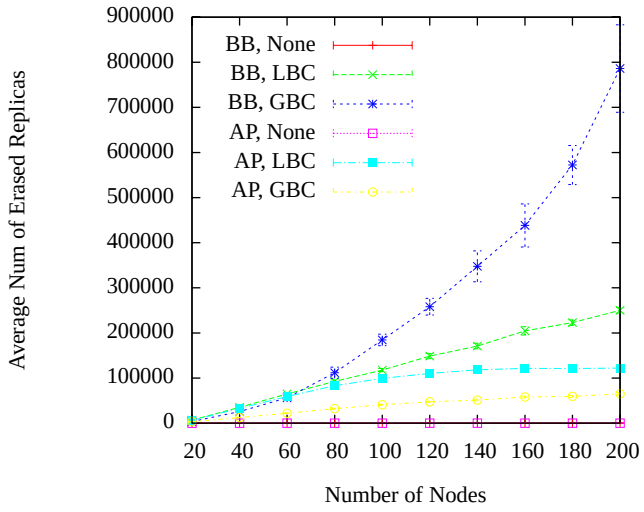

Fig. 15 Erased Replicas - 200 Hoverinfos

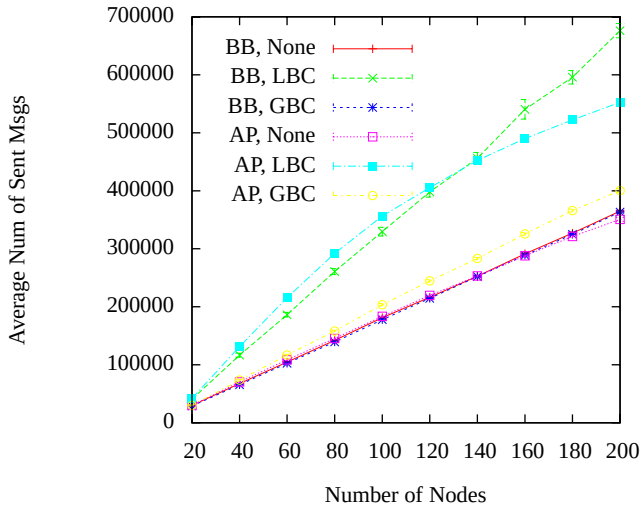

Fig. 16 Messages Complexity - 200 Hoverinfos

LBC caching policy generate more messages than the other cases. The reason of this behaviour is the low availability performances for the algorithms not using the LBC caching in the presence of many pieces of hovering information. For the LBC algorithm, we can also observe that it tends to have slower growing gradient for the AP algorithm compared to that of the BB, which let us suppose that the mes-

sages complexity will get smaller as the number of nodes increases. This shows the scalability of AP with LBC algorithms in terms of network usage.

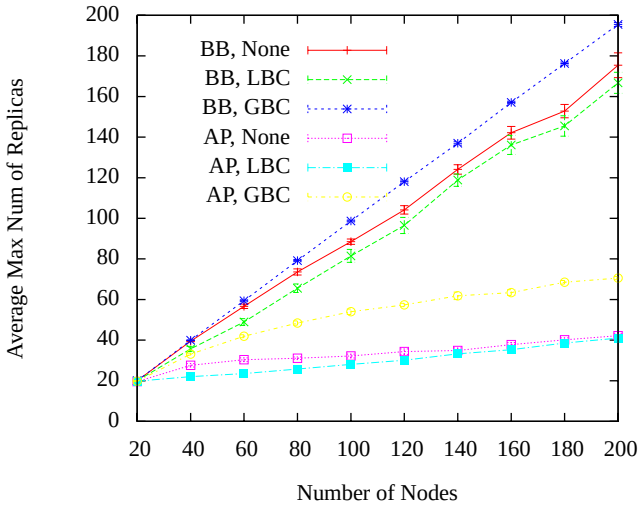

Fig. 17 Replication Complexity - 200 Hoverinfos

Figure 17 shows the replication complexity for all pieces of hovering information in the case of the AP with LBC. This confirms that the AP algorithm limits the number of replicas by concentrating them around the anchor area and not spreading them around all the system as the BB does. We also observe that the LBC caching policy tends to improve the convergence of replicas towards their anchor area.

Figure 18 shows the average of the maximal number of replicas having existed in the system for the AP using the LBC. It is interesting to see that it decreases as the number of pieces of hovering information increases. It means that the buffer resources are evenly shared among the different pieces of hovering information, while the availability still remains at high levels (see Figure 12). We conclude from this, that the AP with LBC succeeds to distribute the network resource in a fair way among all the pieces of hovering information, and that we probably observe an emergent load-balancing of the memory allocated to the different pieces of hovering information.

Finally, Figure 19 shows the average concentration for both algorithms using the different caching policies. We can observe that the LBC policy improves the concentration factor of both algorithms compared to the other caching policies. Particularly, the combination AP and LBC reaches a concentration factor of around 25%.

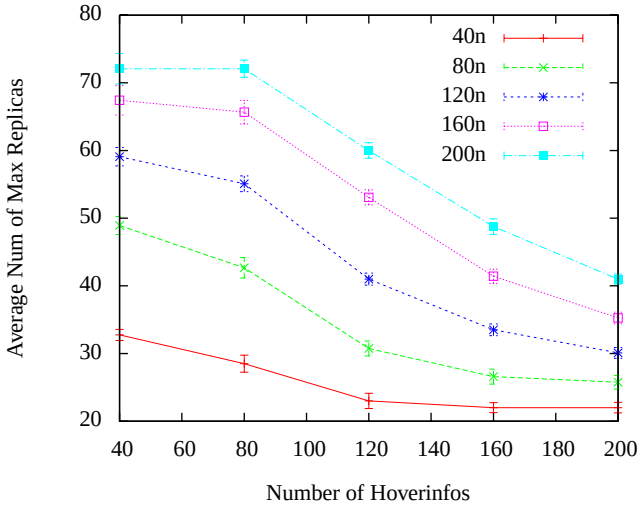

Fig. 18 Replication Complexity - Attractor Point with Location-Based Caching

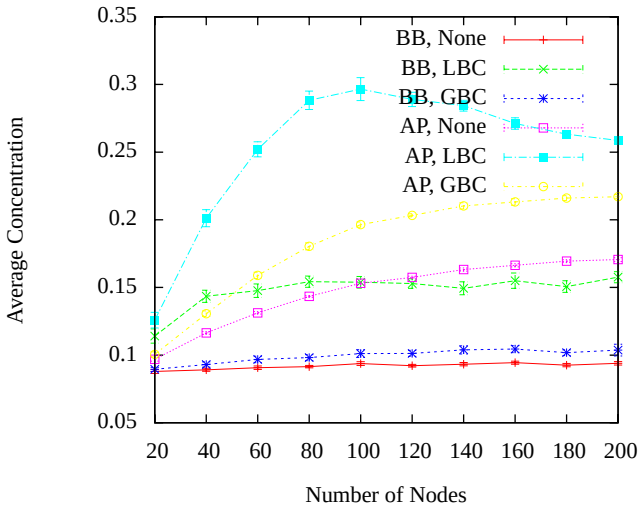

Fig. 19 Concentration - 200 Hoverinfos

6 Related Works

The Virtual Infrastructure project [4, 5, 6, 7] 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. Quorums work as

atomic memory cells and ensure their persistency by replicating their state in neighbouring mobile devices. This notion has been extended to the idea of *virtual mobile nodes* which are state machines having a fixed location or a well-defined trajectory and whose content is also replicated among the nearby mobile devices. Finally, this project provided the notion of a *timed I/O automaton mobile node* where virtual mobile nodes access a clock in order to perform real-time operations. 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 [10] proposes a geographical opportunistic routing algorithm over VANETs (Vehicular Ad Hoc Networks). The algorithm selects appropriate cars for routing some information from a point A to a point B. The choice of the next hop (i.e. the next car) is based on the distance between that cars trajectory and the final destination of the information to route. The planned trajectory generated by a navigation system of the car is used when estimating the relevance of a car to route some information. 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 [11] aims to disseminate traffic information in a network composed by infostations and cars. The system follows the publish/subscribe paradigm. Once a publisher creates some information, a replica is created and propagated all around where the information is relevant. Cars having a replica periodically poll their neighbouring cars, using a broadcast message, to know whether they are interested or not in the replica's information. If some cars reply in an affirmative way the information is sent to them. Based on these periodic pollings, clusters are composed and replicas are removed or propagated to clusters where more subscribers and interested cars are situated. Replicas are also propagated to a randomly chosen car part of the cluster driving in the opposite direction to that of the current host in order to try to keep the information in its relevant area. Cars reply polling with their interests and also their direction. 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 [9, 2] is part of the MoSAIC project and intends to set up a collaborative backup system for mobile devices. The two main issues the authors investigate are: fault- and intrusion-tolerant collaborative backup; and the self-carried reputation and rewards for collaboration. The environment consists of sporadically interconnected and mutually suspicious peer devices having no fixed infrastructure and access to trusted third parties. 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 [14] 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.

7 Conclusion

In this chapter we have defined the notion of hovering information in a formal way and we have defined and simulated the Attractor Point algorithm which intends to keep the information alive and available in its anchor area. This algorithm multicasts hovering information replicas to the nodes that are closer to the anchor location of the information. The performances of this algorithm have been compared to those of a Broadcast-Based algorithm which broadcasts replicas regardless of the proximity or not to the anchor location.

We have also defined and simulated two different caching polices, the Location-Based Caching and the Generation-Based Caching. Their performances have been compared under a scenario containing multiple pieces of hovering information and nodes having a limited amount of memory.

Results show that the Attractor Point algorithm with the Location-Based Caching policy is scalable in terms of the number of pieces of hovering information that the system can support (absorption limits). They also show the emergence of a load-balancing property of the buffer usage which stores replicas in an equilibrated and optimal way as the number of pieces of hovering information increases.

7.1 Future Works

Real mobility patterns We have tested the algorithms under a Random Way Point mobility model and under ideal wireless conditions. This is not characteristic of real world behaviour. We are currently applying the different algorithms to scenarios following real mobility patterns (e.g. crowd mobility patterns in a shopping mall or traffic mobility patterns in a city) with real wireless conditions (e.g. channel interferences or physical obstacles).

Real wireless conditions The simulations performed have been done in an environment where there were neither wireless channel interferences nor physical obstacles. In real world scenarios, these two factors are inherent to wireless communications. It is thus very important to apply the Attractor Point algorithm in a more realistic environment taking in consideration these factors in order to measure the negative drawbacks on the availability performances.

Spatial Memory Service We are currently defining and implementing a distributed memory service, storing and retrieving pieces of hovering information, exploiting available (stationary and mobile) devices as the main storage medium.

Fragmentation and recombination (swarm) In this chapter we have considered atomic information only, but depending on the size of the hovering information (e.g. an image or even a video) it could be fragmented into smaller pieces to fit in multiple nodes' memory. A query of this information will require a recombination mechanism that recovers the different pieces and gets them reassembled to form the original information. We are currently considering this recombination process as a swarm of self-assembling information particles.

Movement speed and direction The current attractor point algorithm takes in consideration the position of the neighbouring nodes only. A significant improvement will be achieved by taking into consideration the speed and direction of the nodes when choosing the nodes that will host replicas.

Coordinates precision The Attractor Point algorithm and the simulations performed do not consider issues related to the precision of geo-localisation service. Precision is an important factor in real world scenario, since it deeply affects the ideal communication range. In order to cope with reality, it is thus important to evaluate and adapt the algorithm in order to take into account problems related to precision

Other simulation environments Results vary when simulation environments change. In order to further validate and compare the results obtained using OMNet++, we will use other simulation environments such as Swarm [3] or attraction fields [12].

[3] http://www.swarm.org

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. Shvartsman, 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. P. Galanter. What is Generative Art? Complexity Theory as a Context for Art Theory. In *Generative Art Conference*, 2003.
9. 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.
10. 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.
11. 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.
12. M. Mamei and F. Zambonelli. *Field-Based Coordination for Pervasive Multiagent Systems*. Springer Series on Agent Technology. Springer Verlag, Berlin, 2005.
13. M. Mamei and F. Zambonelli. Pervasive pheromone-based interaction with rfid tags. *TAAS*, 2(2), 2007.
14. 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.
15. 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.
16. 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.