Hovering Information : Infrastructure-Free Self-Organising Location-Aware Information Dissemination Service

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

[1] Centre Universitaire d'Informatique, University of Geneva,
24 rue Général Dufour, 1211 Geneva 4, Switzerland,
{alfredo.villalba,dimitri.konstantas}@cui.unige.ch
[2] School of Computer Science and Information Systems,
Birkbeck, University of London,
Malet Street, London WC1E 7HX, UK,
dimarzo@dcs.bbk.ac.uk

Abstract. This paper proposes a location-based service for disseminating geo-localised information generated by and aimed at mobile users. The service itself works in a self-organising manner. A piece of hovering information is attached to a geographical point, called the anchor location, and to its vicinity area, called the anchor area. It 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. Previous results involving a single piece of hovering information have shown the interest of the concept. This paper reports on a series of simulations involving multiple pieces of hovering information. Our goal is to investigate the scalability of the technique up to 200 pieces in a small geographic area. Two main replication algorithms for pieces of hovering information are compared, an Attraction Point algorithm and a Broadcast-based one. These replication algorithms are combined with two different caching policies, Location-based and Generation-based, for discarding hovering information pieces from mobile nodes buffer when memory is not enough.

1 Introduction

The last two decades were marked by a rapid evolution of computer and communication related technologies available to the end-users. From the 300MHz processors available in the late 80s, today we are above 4GHz, and from 56Kbps networks (modems), we have today home networks of more than 20Mbps. In a similar way mobile storage capacities available to end-users have gone up from 1MB diskettes to 8GB memory sticks. The average end user today has more wireless network, processing power and storage capacity available to a mobile device, like PDA or smart-phone, than the semi-professional user of the late 80s.

It is thus safe to assume that at the end of the 2010s the average user will have a mobile device with more than 4GHz processing power, 100Mbps wireless network connectivity and more than 1TB of local storage capacity, all supported by powerful power supply, allowing him to have a continuous high bandwidth network connection for periods longer than 24 hours. We can thus also anticipate that the mobile available storage capacity of the end-users will be an important percentage of the fixed storage capacity on the planet. Furthermore we can expect that, mobile devices like phones and PDAs will be equipped with high quality geo-localisation hardware (like GPS and Galileo chips).

Besides these technological advances, we have witnessed during the last few years a new direction taken by the end-users in the creation of information. With the available communication means, end-users are changing their behaviour from information consumers to information producers. More and more information created by end-users is becoming available on the internet, as it is observed by the big success of sites like YouTube and FaceBook. We can thus anticipate that for the next decade end-users will keep on producing even more content, making it available to other users in different forms and under different means.

Considering the above predictions for the next decade, we can sketch a daily scenario for the average user of the late 2010s. The user is equipped with a mobile device through which he accesses location related information that was created by him, friends, or even strangers. All this information is stored primarily into his mobile device and may become available to other users without passing by a mobile operator or a centralized server, instead setting up a mobile ad hoc network and taking advantage of the vast amounts of mobile storage capacities available. In this direction, we have defined the concept of Hovering Information, which provides a promising solution by creating the basis and models for large-scale, location related information management. Instead of accessing location-based information via a wireless network operator, the Hovering Information concept will allow mobile devices to disseminate the information among them, thus relieving the load of the wireless networks. Applications such as stigmergy-based systems, traffic management over vehicle networks or mounting distributed information systems on disaster areas could be implemented using hovering information.

This paper presents the hovering information concept, a replication algorithm allowing single pieces of hovering information to get attracted to their respective geographical related locations, called anchor locations; a broadcast-based algorithm for comparing network and memory usage performances; and two mobile nodes caching policies, Location-based and Generation-based, for discarding pieces of hovering information when memory space is full. A complete formal description of the hovering information model is described in [14].

2 Hovering Information Concept

This section describes the main concepts of a hovering information system: mobile nodes, hovering information; and three main dependability requirements of hovering information: survivability, availability and accessibility.

2.1 Mobile Nodes and Hovering Information

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

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

where id is its mobile node identifier, loc is its current location (a geographic location), $speed$ is its current speed in m/s, dir is its current direction of movement (a geographic vector), r_{comm} is its wireless communication range in meters and $buff$ is its buffer (having a limited size) aimed to store replicas of the pieces of hovering information.

A piece of hovering information is a piece of data whose main goal is to remain stored in an area centred at a specific location called the *anchor location*, and having a radius called the *anchor radius*. A *piece of hovering information* h is defined as a tuple:

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

where id is its hovering information identifier, a is its anchor location (geographic coordinate), r is its anchor radius in meters, n is the mobile node where h is currently hosted (hosting node), $data$ is the data carried by h, $policies$ are the hovering policies of h and $size$ is the size of h in bytes. Policies stand for hovering policies stating how and when a piece of hovering information has to hover.

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. 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.

A hovering information system is composed of mobile nodes and pieces of hovering information. A hovering information system at time t is a snapshot (at time t) of the status of the system. We denote by $\mathcal{N}_t$ the set of mobile nodes at time t. Mobile nodes can change location, new mobile nodes can join the system and 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 1 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 (in three different mobile nodes n_2, n_3 and n_4), while two replicas of h_2 are present in the anchor area of h_2 (in nodes n_2 and n_5). It may happen that a mobile node 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.

2.2 Properties - Requirements

Survivability. A hovering information h is alive at some time t if there is at least one node hosting a replica of this information. The survivability along a period

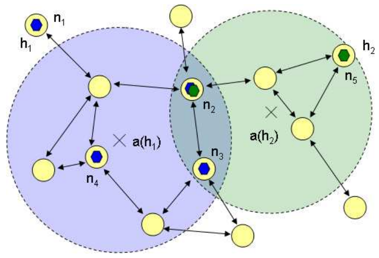


Fig. 1. Hovering Information System at time t

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.

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

Accessibility. 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. The accessibility of a piece of hovering information h is the rate between the area covered by the hovering information's replicas and its anchor area.

The interested reader can refer to [14] for a full set of definitions.

3 Algorithms for Hovering Information

In [15] we studied the performances in terms of availability of a hovering information system containing many replicas of *only one* piece of hovering information. We assumed that each node had a buffer with an unlimited amount of memory for storing replicas. Therefore, the proposed replication algorithms should not have had to cope with buffer overflows problems when a new incoming replica arrived. In this paper, we drop the assumption of unlimited memory and we study a hovering information system containing *multiple* (distinct) hovering information and their respective replicas. Instead of keeping an unlimited buffer size for storing replicas, we limit the size of the buffer. The need for caching policies becomes then important when it is time to insert a new incoming replica.

In this paper, we propose and study two different caching policies: Location-Based Caching (LBC) and Generation-Based Caching (GBC). The LBC policy decision to erase a replica is based on the proximity of that replica to its anchor location and on the portion of the surface of the anchor area covered by the communication area of the node hosting it. The GBC policy takes the decision of removing a replica based on the generation of the replica, removing replicas having the oldest generations.

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

3.1 Safe, Risk and Relevant Areas

In this paper we consider that all pieces of hovering information have the same hovering policies: active replication and hovering in order to stay in the anchor area (for survivability, availability and accessibility reasons), caching when there is no free space to store a replica, 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 node in which the hovering information is currently stored. Therefore, we distinguish three different areas: safe area, risk area and relevant area.

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. This area is defined as the disc having as centre the anchor location and as radius the safe radius (r_{safe}).

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 survive and stay available in the vicinity of the anchor location. This area is defined as the ring having as centre the anchor location and bound by the safe and risk radii (r_{risk}).

The *relevant area* limits the scope of survivability of a piece of hovering information. This area is defined as the disc whose centre is the anchor location and whose radius is the relevant radius (r_{rele}). The *irrelevant area* is all the area outside the relevant area. A piece of hovering information located in the irrelevant area can disappear; it is relieved from survivability goals.

All these radii cope with the following inequality (where r is the anchor radius):

$$r_{safe} < r < r_{risk} < r_{rele}$$

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.

3.2 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 replication algorithms simulating two variants of the replication policies: the Attractor Point and Broadcast-based algorithms. 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.

When replicas consider themselves too far from their anchor area and not able to come back anymore, the cleaning mechanism 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 avoids as well the situation where all nodes have a replica.

Attractor Point Algorithm (AP) 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 other. The number of target nodes composing the multicast group that will receive a replica is defined by the constant k_R (replication factor).

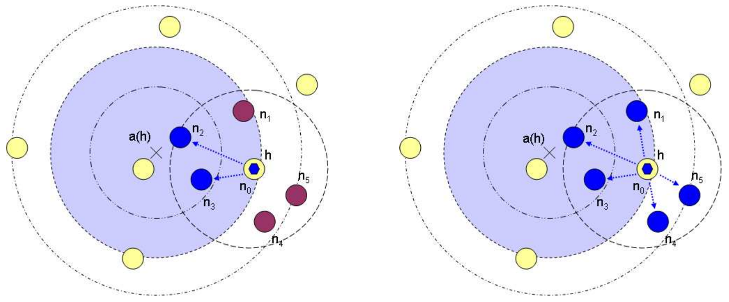
Figure 2(a) illustrates the behaviour of the Attractor Point algorithm. Consider a piece of hovering information h in the risk area. It replicates itself onto the nodes in communication range that are the closest to its anchor location. For a replication factor $k_R = 2$, nodes n_2 and n_3 receive a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

Broadcast-based Algorithm (BB) The Broadcast-based algorithm is triggered periodically (each T_R) for each mobile node. After checking the position of the mobile node pieces of hovering information located in the risk area are replicated and broadcasted onto *all* the nodes in communication range. 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 2(b) illustrates the behaviour of the Broadcast-based 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).

3.3 Caching

In this paper 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 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: Location-Based and Generation-Based Caching. We compare these caching techniques with a simpler



(a) Attractor Point Algorithm (b) Broadcast-based Algorithm

Fig. 2. Replication Algorithms

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).

Location-Based Caching (LBC) 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, the least location relevant replica is chosen from all the replicas stored in buffer of the node. 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. 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. Otherwise, the incoming replica is just discarded. 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.

Generation-Based Caching (GBC) 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 generation number as they are likely more replicas leaving around than a replica having a lower generation number.

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

4 Evaluation

We evaluated the behaviour of the above described replication and caching algorithms under different scenarios by varying the number of pieces of hovering information, each having many replicas of its. We also considered nodes having a limited buffer size to store the different replicas. We have measured the average availability, message complexity, replication complexity, overflows and erased replicas over the total number of pieces of hovering information existing in the system.

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.

4.1 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_{risk}) of 70m, a relevance radius (r_{rele}) 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 store 20 different replicas. The caching algorithm is constantly listening for the arrival of new replicas.

Based on this generic scenario, we defined 5 specific scenarios with varying number of pieces of hovering information: from 40 to 200 nodes, increasing the number of pieces by 40. Each of this scenarios has been investigated with different replication and caching algorithms, and with a different number of nodes.

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).

4.2 Results

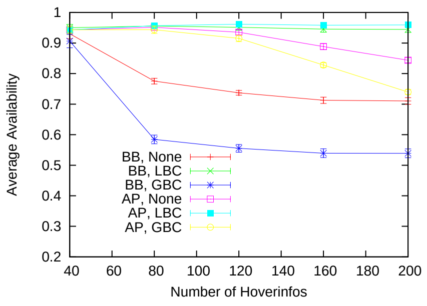


Fig. 3. Availability - 200 Nodes

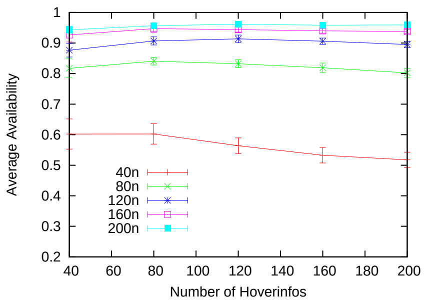


Fig. 4. Availability - AP with LBC

Figure 3 shows the average availability for the AP and BB algorithms using LBC or GBC caching policies, or without using any (None). For this experiment we used 200 nodes. We observe that both algorithms using LBC outperform the cases using GBC or no caching policy (None). We can also notice that the BB algorithm gets worse availability performances compared to AP. This is explained by the nature of the BB algorithm which tends to overload the system with an exponential growing number of replicas. As the buffer size at each node has a limited size of 20 replicas, the overloading causes the buffer resources to become badly shared by the pieces of hovering information, in particular regarding their individual target anchor location. Consequently the availability becomes lower. On the other hand, the AP algorithm controls the replication process by limiting the number of replicas and by focusing on the anchor location. When it is combined with the LBC caching policy, the system becomes scalable keeping high availability rates (around 95%) as the number of pieces of hovering information increases.

Figure 4 depicts the average availability of the AP replication algorithm using the LBC caching policy, under several number 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 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.

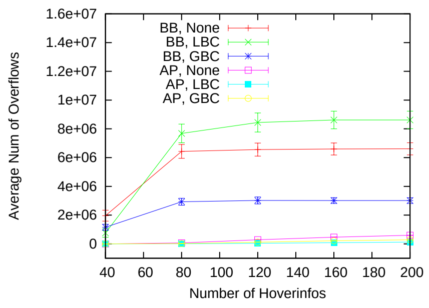


Fig. 5. Overflows - 200 Nodes

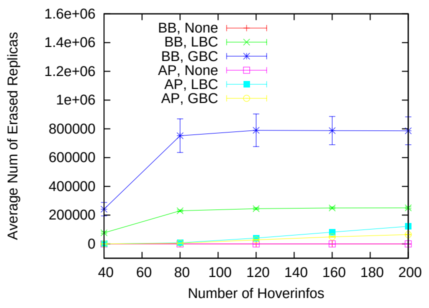


Fig. 6. Erased Replicas - 200 Nodes

Figure 5 shows the average number of overflows generated by the different replication and caching algorithms each time a new incoming replica arrives and there is no free space to store it. We observe that the AP algorithm produces at least ten times less overflows than the BB algorithm. We also observe that the number of overflows of the BB algorithm starts growing fast and then it gets stable after for 80 pieces of hovering information or more, this is again a consequence of the overloading of the system which tends to lose pieces of hovering information by not distributing the buffer resources in a fair way.

Figure 6 illustrates the average number of erased replicas from the buffer of the nodes after an overflow. Again, the BB erases around 10 times much more replicas than the AP algorithm to store a new incoming replica. We also notice that the BB with GBC tends to erase too many replicas compared to the other ones; this 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 algorithms tends to insert and erase replicas permanently.

Figure 7 shows the average number of sent messages for the AP algorithm using the LBC policy under several numbers of nodes. We notice that the message complexity does not grow exponentially. Instead, it grows linearly of even logarithmically with the number of pieces of hovering information. This is a very important issue as the feasibility of these algorithms depends strongly on the messages complexity, especially when we will need to deal with the interference channel for even more realistic network interfaces.

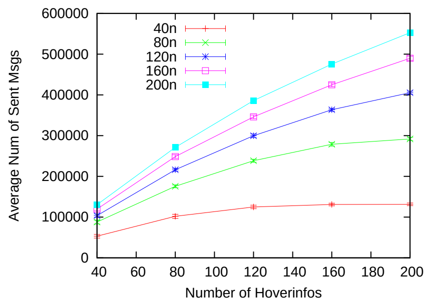


Fig. 7. Messages Complexity - Attractor Point with Location-Based Caching

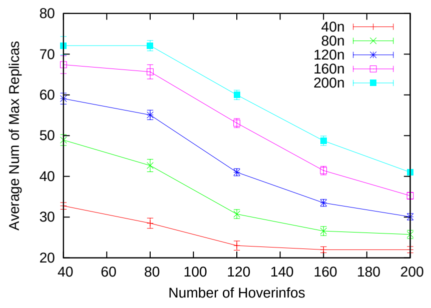


Fig. 8. Replication Complexity - Attractor Point with Location-Based Caching

Finally, figure 8 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 4). 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 (not coded) load-balancing of the memory allocated to the different pieces of hovering information.

5 Related Works

To our knowledge, while there exist other information annotation/dissemination works closely related to the hovering information concept, they do not take the approach of offering a generic infrastructure-free location-aware information dissemination service as hovering information does. The Hovering Data Clouds (HDC) concept [16, 8], which is part of the AutoNomos project, is applied to the specific design of a distributed infrastructure-free car traffic congestion information system. Although HDCs are defined as information entities having properties similar to hovering information, the described algorithms do not consider them as an independent service but as part of the traffic congestion algorithms. The hovering information dissemination service is thought as a service independent from the applications using it. The Ad-Loc system [1] is an infrastructure-free location-aware annotation system and shares similarities to hovering information. However, this approach does not focus on: studying properties such as the critical number of nodes or the absorption limits; or dealing with self-organizing algorithms allowing the information to adapt its behaviour according to the network saturation, the buffers' size, the mobility pattern of nodes or the number of replicas.

The opportunistic spatio-temporal dissemination service over MANETs [11] is a car traffic centred application and does not encompass ideas such as recombination of information. Similarly, works like Epcast [13] and Gossip [3] aim to disseminate information based on epidemic and gossiping spreading models. These works provide an interesting starting point for replication algorithms, but do not offer a solution for ensuring the persistency of the information.

In the domain of location-driven routing over MANETs, we can mention works such as GeoOpss [10], search and query propagation over social networks like PeopleNet [12] and collaborative services such as collaborative backup of the MoSAIC project [9, 2].

Finally, the virtual infrastructure project [4–7] aims to set up a set of virtual nodes having a well-know structure and trajectory over a mobile ad hoc network. These virtual nodes are equipped with a clocked automaton machine which will permit to implement distributed algorithms such as leader election, routing, atomic memory, motion coordination, etc. This approach works on offering a structured abstraction layer of virtual nodes. Hovering information takes a different approach where each piece of hovering information is an autonomous entity responsible for its own survivability exploiting the dynamics of the overlay network to this aim.

6 Conclusion

In this paper we discussed the notion of hovering information, we 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. The performances of this algorithm have been compared to those of a broadcast version.

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 Atrractor 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 optimal way as the number of pieces of hovering information increases.

Concerning future work, 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 will apply 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).

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. A. Datta, S. Quarteroni, and K. Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In *IC-SNW'04, International Conference on Semantics of a Networked World*, LNCS, pages 126–143, 2004.
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. S. P. Fekete, , C. Schmidt, A. Wegener, and S. Fischer. Hovering data clouds for recognizing traffic jams. In *Proceedings 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (IEEE-ISOLA)*, pages 213–218, 2006.
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. 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.
13. S. Scellato, C. Mascolo, M. Musolesi, and V. Latora. Epcast: Controlled dissemination in human-based wireless networks by means of epidemic spreading models. *CoRR*, abs/0711.2780, 2007.
14. A. Villalba Castro, G. Di Marzo Serugendo, and D. Konstantas. Hovering information - self-organising information that finds its own storage. Technical Report BBKCS-07-07, School of Computer Science and Information Systems, Birkbeck, University of London, Nov 2007.
15. 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.
16. A. Wegener, E. M. Schiller, H. Hellbrck, S. P. Fekete, and S. Fischer. Hovering data clouds: A decentralized and self-organizing information system. In *International Workshop on Self-Organizing Systems*, pages 243 – 247, 2006.