Volume 26, issue 6, June 2010
ISSN 0167-739X
ELSEVIER
THE INTERNATIONAL JOURNAL OF
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
GRID COMPUTING AND eSCIENCE
Editor-in-Chief:
Peter Sloot
Available online at www.sciencedirect.com
ScienceDirect

Future Generation Computer Systems 26 (2010) 877–889

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Handling dynamics in diffusive aggregation schemes: An evaporative approach

Nicola Bicocchi *, Marco Mamei, Franco Zambonelli

Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia, Italy

A R T I C L E   I N F O

Article history:
Received 16 July 2009
Received in revised form
4 February 2010
Accepted 10 February 2010
Available online 17 February 2010

Keywords:
Large-size networks
Diffusive aggregation
Gossip-based communication
Sensor networks
Peer-to-peer networks

A B S T R A C T

Distributed computing in large-size dynamic networks often requires the availability at each and every node of globally aggregated information about some overall properties of the network. In this context, traditional broadcasting solutions become inadequate as the number of participating nodes increases. Therefore, aggregation schemes inspired by the physical/biological phenomenon of diffusion have been recently proposed as a simple yet effective alternative to solve the problem. However, diffusive aggregation algorithms require solutions to cope with the dynamics of the network and/or of the values being aggregated solutions, which are typically based on periodic restarts (epoch-based approaches). This paper proposes an original and autonomic solution, relying on coupling diffusive aggregation schemes with the "bio-inspired" mechanism of evaporation. While a gossip-based diffusive communication scheme is used to aggregate values over a network, gradual evaporation of values can be exploited to account for network and value dynamics without requiring periodic restarts. A comparative performance evaluation shows that the evaporative approach is able to manage the dynamism of the values and of the network structure in an effective way: in most situations it leads to more accurate aggregate estimations than epoch-based techniques.

1. Introduction

Controlling and orchestrating global distributed computations in large-size dynamic networks, such as sensor networks [1], P2P overlays [2–4] and grids [5], is particularly challenging. In fact, one has to account for a large number of nodes dynamically joining and leaving the network and changing their local states.

A recurrent distributed computing problem for these kinds of networks is that of making available at each and every node information about some global properties of the network itself (e.g., the average load or the current number of nodes) or of the environment in which the network is deployed (e.g. the average or maximum temperature measured by a wireless sensor network). Many techniques have been developed in the recent past to produce such information (e.g., techniques based on flooding [6] or on spanning trees [7]). However, these appear suitable only for networks of limited size and dynamism. To deal with large-size networks in dynamic scenarios, other directions have to be, and have been, explored.

Protocols inspired by epidemic metaphors are very promising in such a scenario (see [8] for an excellent introduction to the area). In particular, schemes inspired by the physical/biological phenomenon of *diffusion* have been proposed as a simple yet effective solution to propagate local values over a large-size network and to aggregate values over the whole network [9–11], or a portion of it [12]. Several applications of diffusive-based aggregation schemes (typically realized via gossip-based communication [9]) have been already reported in the literature [13,10,14–19].

In diffusive-based aggregation schemes, each node in a network can locally obtain a global picture of some properties of the network, and can exploit such knowledge to go beyond its local viewpoint and adapt/control its activities in a more informed way. However, in dynamic situations, the global representation available at a node is at risk of ageing, and of no longer reflecting the current global state of the network. Indeed, for aggregation algorithms to be effective, they should be able to manage situations in which (i) the values that are being aggregated change over time and (ii) nodes dynamically join and leave the network.

The most widely adopted technique to deal with dynamic situations is the so-called epoch-based approach, relying on periodic restarting of the aggregation process [9]. Periodically, aggregated values (which may no longer reflect the current situation) are invalidated and substituted with those resulting by the execution of a new aggregation process (i.e., of a new "epoch"). Unfortunately, as we will discuss in this paper, epoch-based approaches can induce highly inaccurate and unreliable results in aggregation, other than requiring a careful tuning of internal parameters.

The contribution of this paper is to propose an innovative technique to handle dynamics in diffusive-based aggregation protocol. The inspiration behind our idea is that of pheromone

* Corresponding author.
E-mail addresses: nicola.bicocchi@unimore.it (N. Bicocchi), marco.mamei@unimore.it (M. Mamei), franco.zambonelli@unimore.it (F. Zambonelli).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.02.008

evaporation in ant foraging [20–22]. In ant foraging, the process of collective building of food-nest pheromone trails by ant, is coupled with a process of slow evaporation of pheromone trails. In this way, those trails that are no longer valid tend to disappear, while the still valid ones are continuously re-enforced. This mechanism thus plays a key role in making the overall ant foraging process capable of adapting to dynamic environments. Analogously, our idea—which totally avoids periodic restarts—is that of making aggregated values slowly evaporate. In this way, aged aggregated values that do no longer reflect an up-to-date situation disappear, while those that are still up-to-date will be re-enforced.

Experiments performed on a wide range of situations and simulated network scenarios, together with comparative evaluations against two variants of the epoch-based technique, show that the evaporative approach is able to handle the dynamism of the values measured over the network in an effective way, and in the most of the cases is more accurate than epoch-based techniques. In particular, the evaporative solution is to be preferred either in very large networks or whenever periodic restarts are to be avoided.

The remainder of this paper is organized as follows. Section 2 introduces diffusive aggregation protocols and the issues of handling dynamic scenarios. Section 3 describes two variants of the epoch-based technique based on periodic restarts, and outlines their limitations. Section 4 introduces and details our original evaporative approach. Section 5 describes the network setup used in our experiments and evaluates, also in a comparative way, epoch-based approaches and our evaporative approach. Section 6 presents related work in the area. Section 7 concludes and illustrates some future work.

2. Diffusive aggregation

In this section, we introduce the gossip-based communication scheme that is typically adopted in diffusive-based data aggregation, and then we discuss the typical problems arising in the presence of dynamic situations.

2.1. The basic algorithm

The three approaches we describe and compare in this paper share the same diffusive aggregation algorithms and rely on the same gossip-based communication (as from [9]). The protocol abstracts from any specific network technologies and topologies and is not affected by them.

Let us consider a generic network of n nodes in which each of the nodes is connected with a small ($\ll n$) number of other nodes (neighbors). The gossip-based communication protocol is based on the ''push–pull'' scheme, as illustrated in Fig. 1. Time is considered divided into slices of length t. Each node p executes two different threads. At every slice, an active thread starts an information exchange with a random subset of its neighbors q by sending them a message containing the local state s_p and waiting for a response with the remote state s_q. s_p and s_q are the aggregated values that the algorithm is computing on the p and q nodes respectively. A passive thread waits for messages sent by an initiator and replies with the local state. The term *push–pull* refers to the fact that each information exchange is performed in a symmetric manner: both participants send and receive their states. This is a general scheme and the local state s_p could represent whatever property of the node p or whatever value locally measured by it.

The method GETNEIGHBORS(ω) can be thought of as an underlying service to the gossip-based protocol, which is normally (but not necessarily) implemented by sampling a locally available set of neighbors. We assume that GETNEIGHBORS(ω) returns a uniform random sample over the entire set of neighbors. Moreover the parameter ω can be used to specify how many neighbors have to be returned. In particular, with $\omega = 1$ it will return all the available neighboring nodes, while with $\omega = 1/\text{neighbors}$ only one neighbor will be returned (as in [9]). It is convenient to describe the GETNEIGHBORS(ω) method as a separate service in order to hide the physical differences between networks. For example, over an IP network, neighbors are those nodes connected to the same collision domain. In a wireless sensor network, neighbors are those nodes with a relative physical distance lower than their wireless radio range. In P2P networks, neighbors are those nodes connected in the overlay network.

```
Active Thread

do exactly once in each consecutive
t time units at randomly picked time:
    $q[] \leftarrow$ GETNEIGHBORS($\omega$)
    foreach $q$:
        send $s_p$ to $q$
        $s_q \leftarrow$ receive($q$)
        $s_p \leftarrow$ UPDATE($s_p, s_q$)

Passive Thread

do forever:
    $s_q \leftarrow$ receive($*$)
    send $s_p$ to sender($s_q$)
    $s_p \leftarrow$ UPDATE($s_p, s_q$)
```

Fig. 1. The basic gossip-based protocol for diffusive aggregation rely on a ''push–pull'' scheme. Time is considered divided into slices of length t. Each node p executes two different threads. At every slice, the active thread starts an information exchange with a random subset of its neighbors q by sending them a message containing the local state s_p and waiting for a response with the remote state s_q (such a state representing the value being aggregated). The passive thread waits for messages sent by an initiator and replies with the local state.

The method UPDATE computes a new local state based on the current local state and on the remote state received during the information exchange. It is within the UPDATE method, in particular, that the diffusive aggregation process takes place. The output of UPDATE and the semantics of the node state depend on the specific aggregation function being implemented by the protocol. Fig. 2 reports three exemplary instances of the UPDATE method for the minimum, maximum, and average aggregation functions.

This gossip-based communication scheme on which diffusive aggregation relies acts as a sort of continuous background activity. It tends to impose a predefined, tunable, and uniform load, to the system. Each node executes the same amount of operations. Shorter t or higher ω speed up the convergence of the algorithm and increase the number of exchanged messages. Therefore, every user can select the most suitable trade off between accuracy and communication costs over time. Incidentally, on energy constrained networks like sensor networks, communication costs are highly relevant because they negatively affect the lifespan of the system. Thus, tuning the communication costs is useful in many practical settings.

2.2. Handling dynamics

The generic protocol described so far is not adaptive: aggregation takes into account neither the dynamism of the network nor

```
UPDATEMIN $(s_p, s_q)$:
        return $min\,(s_p, s_q)$

UPDATEMAX $(s_p, s_q)$:
        return $max\,(s_p, s_q)$

UPDATEAVG $(s_p, s_q)$:
        return $(s_p + s_q)/2$
```

Fig. 2. The UPDATE method for three different aggregation functions (i.e. minimum, maximum and average).

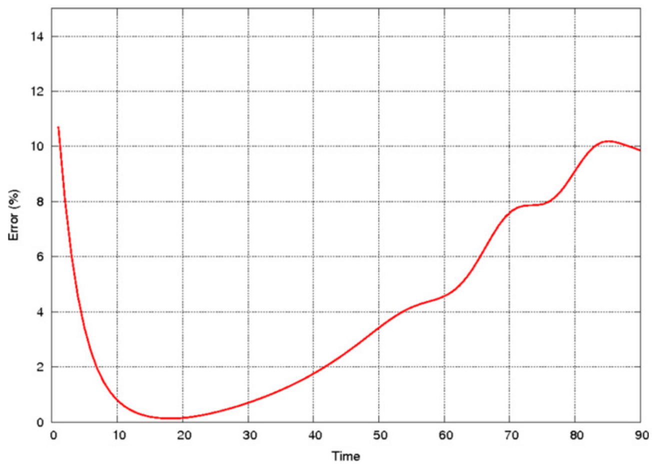

Fig. 3. The average error rate of a diffusive aggregation scheme without any mechanism to handle environment dynamics.

the dynamic changes in the values being aggregated that can take place while the aggregation process is ongoing.

To clarify, let us assume that each node in a network holds a numeric value, which can characterize some aspects of the node or its environment (for example, the load at the node, its storage space, or the atmospheric pressure sensed by it). The goal of a diffusive aggregation is to continuously provide the nodes with an up-to-date estimate of an aggregate function, computed over the values held by the set of network nodes. However, in a realistic scenario, values measured by the nodes may change over time. Moreover, nodes continuously join and leave the network. From this point of view, changes of the values held by the nodes or changes of the network itself (e.g. nodes leaving and joining) could be considered the same. For example a node that suddenly starts measuring a *null* value can be considered as a node that left the network (this consideration is substantiated in the evaluation section). Thus, let us sketch how such forms of dynamism can affect the accuracy of the aggregation process in the absence of any mechanisms to handle it.

In particular, let us qualitatively show what can be the typical situation occurring during the execution of a diffusive aggregation scheme devoted to aggregate the global average of a scalar value sensed/measured at each node of the network. Let avg_p^t be the average computed by node p at time t using the mechanisms illustrated in Fig. 2, and let ravg^t be the ground-truth average value. We can define the average error of the aggregation as $E^t = \frac{1}{n}\sum_{p=0}^{n} \frac{\max(\mathrm{avg}_p^t)-\mathrm{ravg}^t}{\mathrm{ravg}^t}$, where n is the number of nodes in the network.

Fig. 3 plots the typical behavior of such error over time, in the presence of dynamism of the value being averaged.

Starting from an initial state in which the nodes do not yet have any accurate estimation of the global value, the error quickly decreases as the algorithm converges and the nodes acquire a correct local estimation of the aggregated value. Nevertheless, after some iterations and while the local situation at nodes changes because of some dynamic phenomenon (e.g., the values at some nodes diminish), the average error starts to increase. That is, the aggregated value locally estimated by nodes does no longer reflect reality and the error goes out of any control. This happens because there is no mechanism to keep aggregated data updated.

It can be shown (and it is quite intuitive indeed, due to the cumulative nature of aggregation) that aggregation algorithms do not experience problems if executed on a fixed or growing number of nodes measuring steady phenomena. Instead, both the cases in which the network shrinks and in which the values to be aggregated change are more complex to be managed. In fact, in both these situations, the aggregated values may no longer be valid and the cumulative nature of aggregation does not enable them to be properly updated. For example if a node holding a very high value leaves the network, the aggregated values (e.g. maximum or average) propagated to every node will continue accounting for the existence of such a high value.

Considering that all natural and social processes involve changes, and that most modern networks are of an inherent dynamic nature, techniques are required to manage this recurrent situation.

3. Epoch-based approach

To solve the problem and make aggregation schemes able to handle dynamics, the simplest solution is to implement periodic restarts of the scheme [9], i.e., to realize a so-called epoch-based aggregation.

3.1. Basic algorithm

Starting from the algorithm proposed in Fig. 1, we present here the simplest possible form of epoch-based diffusive aggregation, to which we refer as GEA (Gossip-based Epoch Aggregator).

GEA is realized via a very simple mechanism: each node executes the protocol for a predefined number of cycles, denoted as γ, depending on the required accuracy of the output and on the convergence time of the aggregation process. On this base, we divide the protocol execution into consecutive epochs of length $\delta = \gamma t$ (where t is the cycle length), and start a new instance of the protocol at each epoch. That is, at the start of a new epoch, aggregated values are reset to the local value of the node and start being re-aggregated again.

The protocol requires messages to be tagged with an epoch identifier to ensure synchronization. When a node receives message with a new epoch identifier, it understands that a new epoch has been initiated by other nodes in the network and immediately joins the new epoch in its turn.

Fig. 5 shows the basic gossip-based communication scheme presented in the previous section, yet modified with the introduction of the *epochTime* variable to divide the execution into different time slices.

Fig. 4 (to be compared against 3 and referring to similar conditions) shows the trend of the average error during the execution of an epoch-based aggregation scheme. The picture compares also GEA with other two algorithms (i.e., EVP and GEAopt) described below. This highlights an important drawback of GEA, independently of the chosen *epochTime*. Every time an epoch ends and a new epoch starts, locally aggregated values are reset to the node's local value. As a consequence, the average error suddenly increases up, and a portion of the next epoch is needed to make the algorithm converge again. It is clear that the low stability of the error rate is not a desirable property and, under several circumstances, could be unacceptable.

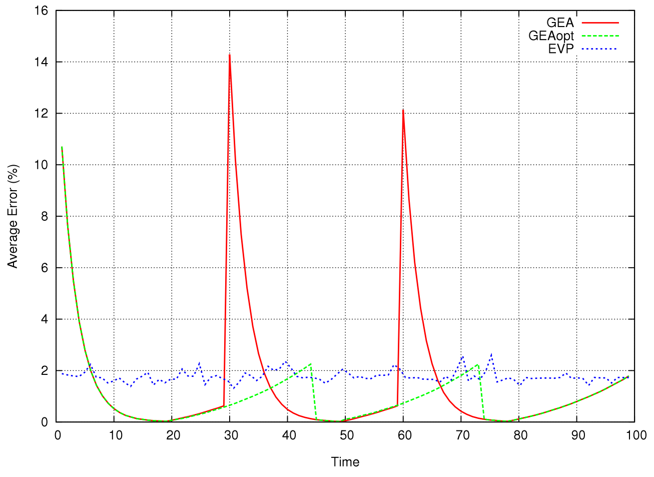

Fig. 4. The average error rate of a diffusive aggregation scheme produced by our original evaporation-based approach (EVP) and two epoch-based approaches (GEA, GEAopt).

```
Active Thread

$epochTime = 0$
    do exactly once in each consecutive
    $t$ time units at randomly picked time:
        $q[] \leftarrow$ GETNEIGHBOR($\omega$)
      foreach $q$:
          send $s_p$ to $q$
            $s_q \leftarrow$ receive($q$)
            $s_p \leftarrow$ UPDATE($s_p, s_q$)
      if $epochTime >= \gamma t$:
          RESET ($s_p$)
```

Fig. 5. The GEA algorithm main body.

3.2. *Optimized algorithm*

We have outlined the biggest drawback of GEA: every time an epoch ends, locally aggregated values are reset to the nodes' local values, causing spikes in the estimated values that, of course, are better to be avoided. For this reason, a modified (optimized) version of GEA is usually adopted, we hereby refer to this as GEAopt.

In GEAopt, nodes participate to two epochs simultaneously, and: (i) nodes consider as currently valid aggregated values those related to an epoch that is assumed to already have properly converged; (ii) at the same time, nodes start concurrently computing new aggregated data over a new concurrently executing epoch. Once the new epoch of aggregation is assumed to have in its turn produced properly aggregated updated values, the data of the older aggregation epoch can be dismissed, nodes can start answering queries with the newly computed aggregated data. A further concurrent epoch can be launched to start computing the next-new aggregated values, and so on. It is worth emphasizing that the communication costs in GEAopt are not higher than in the basic GEA approach, because different epochs are computed in parallel using the same data exchanged in GEA.

This mechanism enforced in GEAopt can greatly reduce error spikes and thus produces better performances than GEA. However, as it can be seen from Fig. 4, the issue of having a great variability in the average errors produced still exists. In fact, while the new epoch is being aggregating new data, the old epoch reports on aggregated data which, in the presence of dynamics, tends to be increasingly inaccurate. Also, the GEAopt approach introduces the issue of properly evaluating the time in which the old epoch should be dismissed and the new one should step up. In our experiment, we choose a static policy in which GEAopt replies with values aggregated over the new epoch when it has accomplished more than half of its length.

4. The evaporative approach

The two above proposed aggregation schemes share the need of periodic restarts. This keeps them fairly simple and efficient. However, in several circumstances, a continuous approach never requiring to be restarted can be useful. For example, over very large networks where the convergence time could be high, an approach able to avoid periodic restarts and, thus, error spikes, could be preferred.

From a conceptual viewpoint, we can think at further improving GEAopt with the existence of multiple concurrent epochs instead of only two ones. In this way, at any given time, one can be sure that an epoch exists whose values are close to convergence. Yet, this leaves open the issue of deciding which—among the many concurrent epochs—is the best one. We overcome this limitation by proposing a bio-inspired mechanism able to mimic the results that would be achievable with a very high, almost infinite, number of parallel epochs, yet avoiding the need of selecting the best one and, in the end, avoiding periodic restarts at all.

The basic principle behind our idea gets its inspiration from one specific mechanism exploited in the phenomenon of ants' foraging [20,22,23,21]. Ants are able to collectively find paths to food by having "happy" ants that have found food spread pheromone trails that lead to the food source. Until such paths are valid (i.e., until they effectively lead to a non-empty food source), paths are implicitly re-enforced by additional "happy" ants that continue walking on it by spreading additional pheromones. However, such a process would not be adaptive if pheromone trails would endlessly cumulate: in the case the food finishes (or in the case an obstacle makes the paths uneasy) the cumulated pheromones on the trail would continue leading ants to no longer useful places. To most extents, this is also what happens in diffusive aggregation: values cumulated in the aggregation continue affecting the aggregation results even when no longer reflecting the actual values of nodes.

The mechanism that makes ant foraging adaptive is that of pheromone evaporation: pheromones' trails, unless properly re-enforced, tend to slowly yet continuously evaporate, such that bad trails eventually disappear. Accordingly, it is possible to think at exploiting evaporation in diffusive aggregation by making the values being aggregated "evaporate", thus letting the obsolete ones disappear while continuing reinforcing the up-to-date ones.

In practice, in our approach, the aggregated values at a node are slowly (compared to the convergence time of the aggregation algorithms) moved towards the local values of the node. In this way, the weight of those data cumulated by the aggregation algorithm will gradually diminish, unless properly re-enforced. The result (as from Fig. 4) is that the estimate error of an aggregated value is much less variable, slightly oscillating around a small error value, yet never exhibiting the spikes typical of epoch-based approaches.

As an example, consider the case of a network aggregating the maximum of a local property (illustrated in Fig. 6). Assume that each node has already locally available its own maximum estimator. Now, have each node slightly "evaporates" such a value by making it diminish to approach the local value. If the node holding the actual maximum does not change, a node will receive again the maximum value undoing the evaporation effects. Instead, if the value measured by the node holding the maximum decreases, evaporation will stabilize the new maximum at each node, after

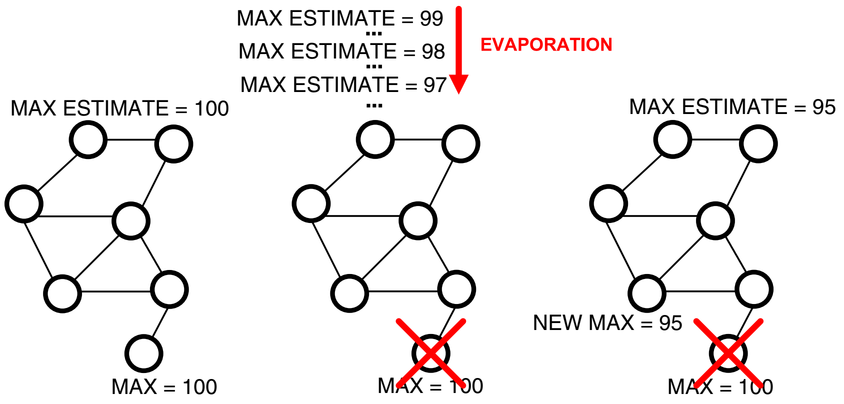

Fig. 6. The evaporative approach at work. (left) Diffusive aggregation has properly converged, and nodes correctly estimate that the global maximum is 100. (center) The node containing the maximum is disconnected from the network. The estimated maximum at the other nodes slowly evaporates, thus reducing the estimate. (right) The diffusive process eventually makes nodes acquire a correct estimate of the new maximum.

```
UPDATEMIN($s_p, s_q$):
    $m_p = min(s_p, s_q)$
    return $min(m_p + \Delta, s_p)$

UPDATEMAX($s_p, s_q$):
    $M_p = max(s_p, s_q)$
    return $max(M_p - \Delta, s_p)$

UPDATEAVG($s_p, s_q$):
    $w_p = \frac{w_p + w_q + s_q - s_p}{2}$
    if $(abs(w_p) < \Delta) w_p = 0$
    if $(w_p < -\Delta) w_p = w_p + \Delta$
    if $(w_p > \Delta) w_p = w_p - \Delta$
    return $w_p$
```

Fig. 7. The UPDATE* functions of the evaporative approach, as used to compute the aggregated values of *minimum*, *maximum* and *average*.

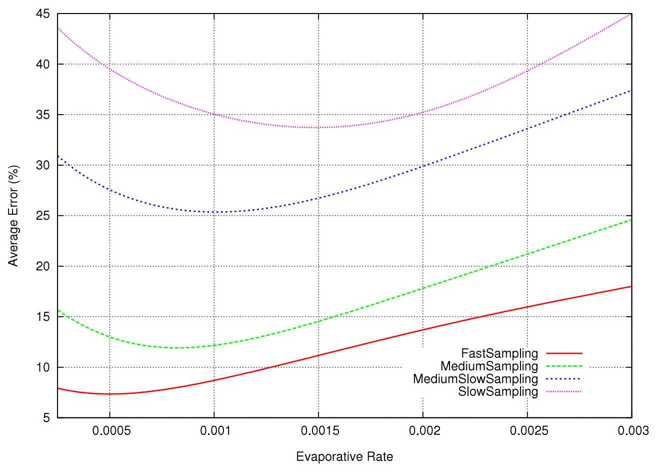

Fig. 8. The effects of the evaporative rate (Δ) over the EVP approach. For each possible value of t we run several simulations using different values of Δ ($n = 1000$, $\omega = 1$). It is clear from the graph that for every t exists an optimum value of Δ which produces the lowest error. Moreover, it is clear that to compensate low sampling frequencies higher values of Δ are required.

proper evaporation. Similar considerations can be applied to many other aggregation functions (i.e., to the whole class of order and duplicate insensitive aggregation functions [6]).

Another very relevant aggregation function, worth exemplifying, is the average. The algorithm we propose for calculating the average in a fully decentralized way apparently only slightly differs from that discussed in [9]. However, it has the advantage of making it possible to tolerate the evaporative approach. The algorithm works as follows: let p and q be two neighbor sensors and $s_p(t)$ and $s_q(t)$ be the values sensed by them, respectively at time t. Then define the average estimator avg as:

$$\text{avg}_p = s_p(t) + w_p(t); \qquad w_p(0) = 0$$
$$\text{avg}_q = s_q(t) + w_q(t); \qquad w_p(0) = 0.$$

In other words, the average estimator is the sum of the local value measured by the node and a second value w which represents the difference between the local value and the estimated average. w is the term subject to evaporation. Decoupling w from s allows the evaporation process: positive values of w has to be diminished, negative increased. At every iteration we aim to update the value of w to let each node converge to the right value of avg. In particular:

$$w_p(t) + w_q(t) = w_p(t+1) + w_q(t+1)$$
$$s_p(t) + w_p(t+1) = s_q(t) + w_q(t+1).$$

Resolving in $w_p(t+1)$ and $w_q(t+1)$ it leads to:

$$w_p(t+1) = (w_p(t) + w_q(t) + s_q(t) - s_p(t))/2$$
$$w_q(t+1) = (w_p(t) + w_q(t) + s_p(t) - s_q(t))/2.$$

It can be shown that by asynchronously applying the above calculation for subsequent couples of nodes in the system, whatever the order is, makes the average estimator avg at each node converge toward the global average of the value s in the region. And that the convergence is not undermined by dynamic changes in the values s of the nodes of the network. Most importantly, this new approach maintains the general diffusion scheme presented in Fig. 1 but exploits a particular logic into the function UPDATE.

In Fig. 7, we reported three different UPDATE functions to manage *minimum*, *maximum* and *average* estimation using our evaporative technique (EVP, for short).

Clearly EVP performances are also influenced by the evaporative rate Δ (the only parameter of this approach). For each possible value of t we run several simulation using different values of Δ to assess the effect of this parameter. Fig. 8 shows the relation between the average error rate and Δ.

The graph shows different curves for different sampling frequencies. Given a phenomenon which would be properly monitored with a known sampling period T, we conducted experiments using: $t = T/3$ (*fast* sampling), $t = T$ (*medium* sampling), $t = 2T$ (*medium–slow* sampling), $t = 3T$ (*slow* sampling). From the algorithm viewpoint, *fast* sampling implies the perception of a slow phenomenon (intuitively because the algorithm is faster than the phenomenon being observed), *slow* sampling implies the perception of a fast one (intuitively because the algorithm is slower than the phenomenon being observed).

It is clear from the graph that, for every t, an optimum value of Δ which produces the lowest error exists. Moreover it is clear that, to compensate low sampling frequencies higher values of Δ are required. This happens because a bigger time slot between two consecutive samplings allows bigger changes into the environment. The more the values' fluctuations increase, the more Δ should be high to follow them. In particular this property could be the starting point to apply the evaporative mechanism to systems which have to deliver an autonomic behavior by self-adapting Δ. However, it is also interesting to note that the performances of EVP are not that greatly dependent over the value of Δ chosen. In fact, even if EVP is not able to self-adapt the values of Δ, changes in its value do not produce much greater errors than the optimal value.

5. Performance evaluation

5.1. Network and experiments setup

Diffusive aggregation mechanisms can be used in several different network scenarios, ranging from wireless sensor networks to large-scale peer-to-peer systems. To analyze the behavior of the proposed mechanism under different conditions, we modelled 2 exemplary network setups: (i) one in which nodes can be connected only if they are in physical proximity (this resembles wireless sensor network scenarios), (ii) one in which long-range links exist between nodes (resembling P2P networks).

To test the behavior of the system, in the context of large-scale sensor network, we simulated amorphous spatial network topologies in which nodes are connected on average with k neighbor nodes. Nodes have been positioned in a given area and connected to their $k + N(0, \sigma^2)$ closest neighbors. $N(0, \sigma^2)$ is a 0-mean Gaussian distribution with variance $\sigma^2 = k$. This kind of network is a good match for large-scale sensor network topologies where links can be established only between neighbor nodes (see Fig. 9-left).

In addition, we analyzed network changes departing from amorphous topologies and introducing long-distance links in the network. Once these links are introduced, the network model deviates from prototypical wireless sensor networks. The obtained networks are examples of the Watts–Strogatz model [24] and can model (in a simplistic way) peer-to-peer systems, in which small-world phenomena arise. More in detail, the Watts–Strogatz model prescribes that each link of the network is randomly rewired with probability α. Rewiring an edge at node p means removing that edge and adding a new edge connecting p to another node picked at random. When $\alpha = 0$, the network remains unchanged, while when $\alpha = 1$, all edges are rewired, generating a random graph (see Fig. 9-right). For intermediate values of α, the structure of the graph lies between these two extreme cases: complete order and complete disorder. Considering that with $\alpha > 0.04$ the network fundamental properties do not change significantly (already exhibiting random network properties), we reported only the results obtained with $\alpha < 0.04$.

As already discussed in [9], the resulting topology can effectively model the properties of structured peer-to-peer overlay networks (e.g., Chord [25]), and of unstructured peer-to-peer networks (e.g., Gnutella [26]) with the limitation of presenting small-world, but not scale-free properties [27].

More in detail, to measure the performances of the studied approaches under different conditions, we performed several experiments varying 5 main parameters: 3 related to the network and 2 related to the algorithms itself. In particular we evaluated:

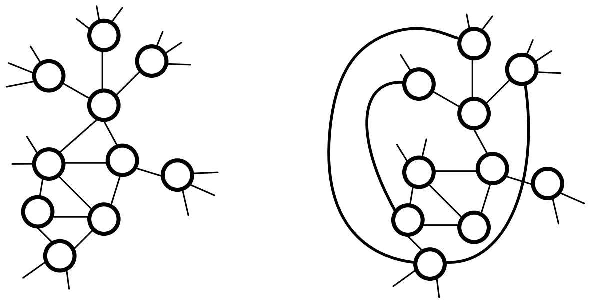
Fig. 9. Network models. (left) Spatial amorphous network modelling a sensor network. (right) Amorphous network with long-range links modelling P2P networks.

- Network-related parameters:
  1. The average number of neighbors (k) of the network's nodes. We investigated networks with an average k between 2 and 16. This network parameter is also useful to analyze the algorithm's parameter (ω). ω represents the fraction of the k-neighbors that is actually considered when gossiping data (see GETNEIGHBORS function in Section 2.1). So, $\omega = 1.0$ means that each node exchanges data with *all* of its k-neighbors. $\omega = 0.2$ means that each node interacts with only the 20% of its k-neighbors. It is clear that the effects of changing ω can be analyzed and understood by changing k. For example, the same overall results can be obtained by setting $\omega = 1$ and $k = 4$ or $\omega = 0.5$ and $k = 8$.
  2. The rewiring percent (α) measuring the presence of long-range links and the emergence of random network's features. We investigated values between 0 and 0.04, to evaluate how the behavior of the algorithm changes while moving from spatial networks to P2P networks.
  3. The number of nodes (n), ranging from 10 up to 10 000 nodes to evaluate the scalability.
- Algorithm-related parameters:
  1. The algorithm's sampling period (t). We were interested in the effects of the speed of the observed phenomenon over the aggregation performances. In particular, given a phenomenon which needs a known sampling period T to be properly monitored, we conducted experiments using: $t = T/3$ (*fast* sampling), $t = T$ (*medium* sampling), $t = 2T$ (*medium–slow* sampling), $t = 3T$ (*slow* sampling). From the algorithm viewpoint, *fast* sampling implies the perception of a slow phenomenon, *slow* sampling implies the perception of a fast one and so on.
  2. The algorithm's neighborhood density (ω) ranging between 0.2 and 1.0. Given our experiments' setup, this range is enough to understand the general effects of this parameter. As discussed above, we do not show any explicit graph about this parameter because of the effects of changing ω can be analyzed and understood by changing k.

For the sake of clarity, in all the experiments with epoch-based approaches we used $\gamma = 50$, which is a proper value for the proposed experiments.

While our experiments mostly deal with dynamism at the level of the values being aggregated, in this paper, we mainly focus on static network topologies (neighbors do not change over time). However, as already anticipated in Section 2.2, the effects of network dynamics on the described algorithms can be easily simulated. In the context of this work, changes of the values held by the nodes or changes of the network itself (e.g., nodes leaving and joining) could be assimilated: a node that suddenly starts measuring a *null* value and that no longer participate in the gossip scheme can be considered as a node that left the network. Accordingly, we developed some experiments in which nodes

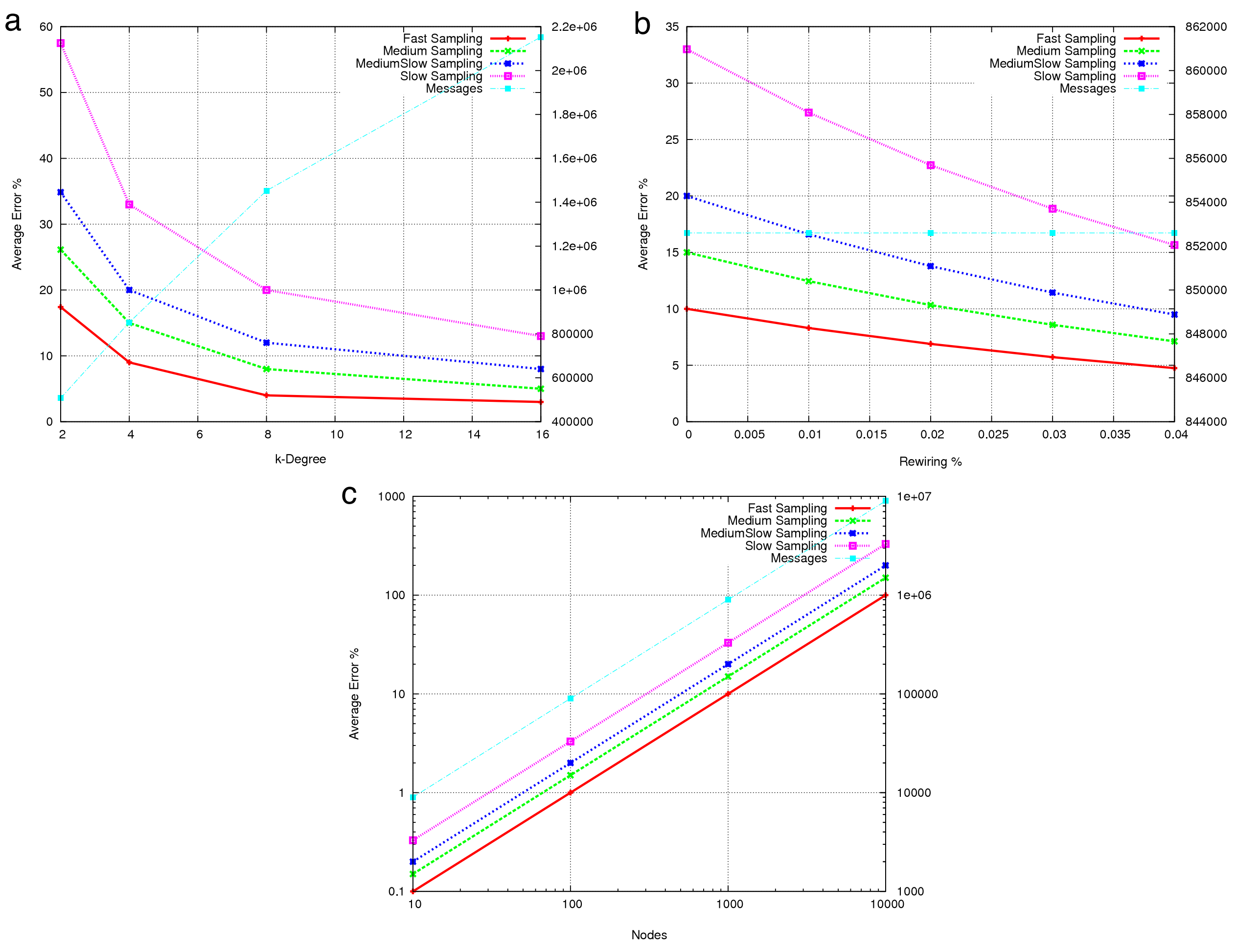

Fig. 10. GEA performances varying network (a) k degree, (b) rewiring factor and (c) size.

provide *null* values with a certain periodicity so as to test the behavior of our algorithms in the presence of network churn (see Fig. 14).

In all the experiments being performed, the local property observed at every node changes over time following a periodic pattern: that is, the behavior of the algorithm in computing global aggregate properties has been evaluated while the values were dynamically changing.

More in detail, we divided the nodes in the network in two groups. Nodes in the first group sense a local property ranging from a value of 0 to 255. Each node starts from a random value in the interval. The value increases by 1 every T simulation cycles until reaching 255. Then it decreases by 1 until reaching 0, and so on. Accordingly nodes sense an oscillating local property. Nodes in the second group senses a similar oscillating property ranging from -255 to 0.

Every experiment has been conducted over a simulation of 250 iterations. Every point in Figs. 10–12 represents results averaged over 25 simulation runs. In all the experiments we fixed all the parameters except the one under investigation.

To measure the effectiveness of the approaches, we decided to use the already introduced average estimator. Since the goal of aggregation is to compute at each node an accurate estimate of a global property, for each experiment, we averaged over all the iterations the distance between the highest estimated average by all the nodes and the actual average (i.e. the worst case). Defined the average estimated by the node p at the iteration i as avg_p^i, and the actual average as ravg^i, we compute the error as:

$$E = \frac{1}{I}\sum_{i=0}^{I}\frac{\max(\text{avg}^i) - \text{ravg}^i}{\text{ravg}^i}$$

where I is the total number of iterations.

We emphasize that we also evaluated the effects of choosing a different error estimator (i.e. the average case). Experimental data suggested that the qualitative trends do not change. Because of this, all the results presented in the following have been obtained using only E.

5.2. Experimental results

Figs. 10–12 summarize GEA, GEAopt and EVP performances respectively. The left vertical axis represents the average error, while the right vertical axis represents the number of messages being exchanged.

The three (a) graphs ($n = 1000, \alpha = 0, C = 0$) show that, for all the three approaches and regardless t, the average error tends to decrease increasing k because of the number of diffusion interactions increases as well. More interactions produce an higher convergence speed. Obviously, also the number of messages being exchanged, and than the communication costs, increases with k.

It is again worth emphasizing that this graph can be interpreted as a graph measuring the effect of the algorithmic parameter ω. The same overall results can be obtained by setting $\omega = 1$ and $k = 4$ or $\omega = 0.5$ and $k = 8$. That is, the effect of doubling

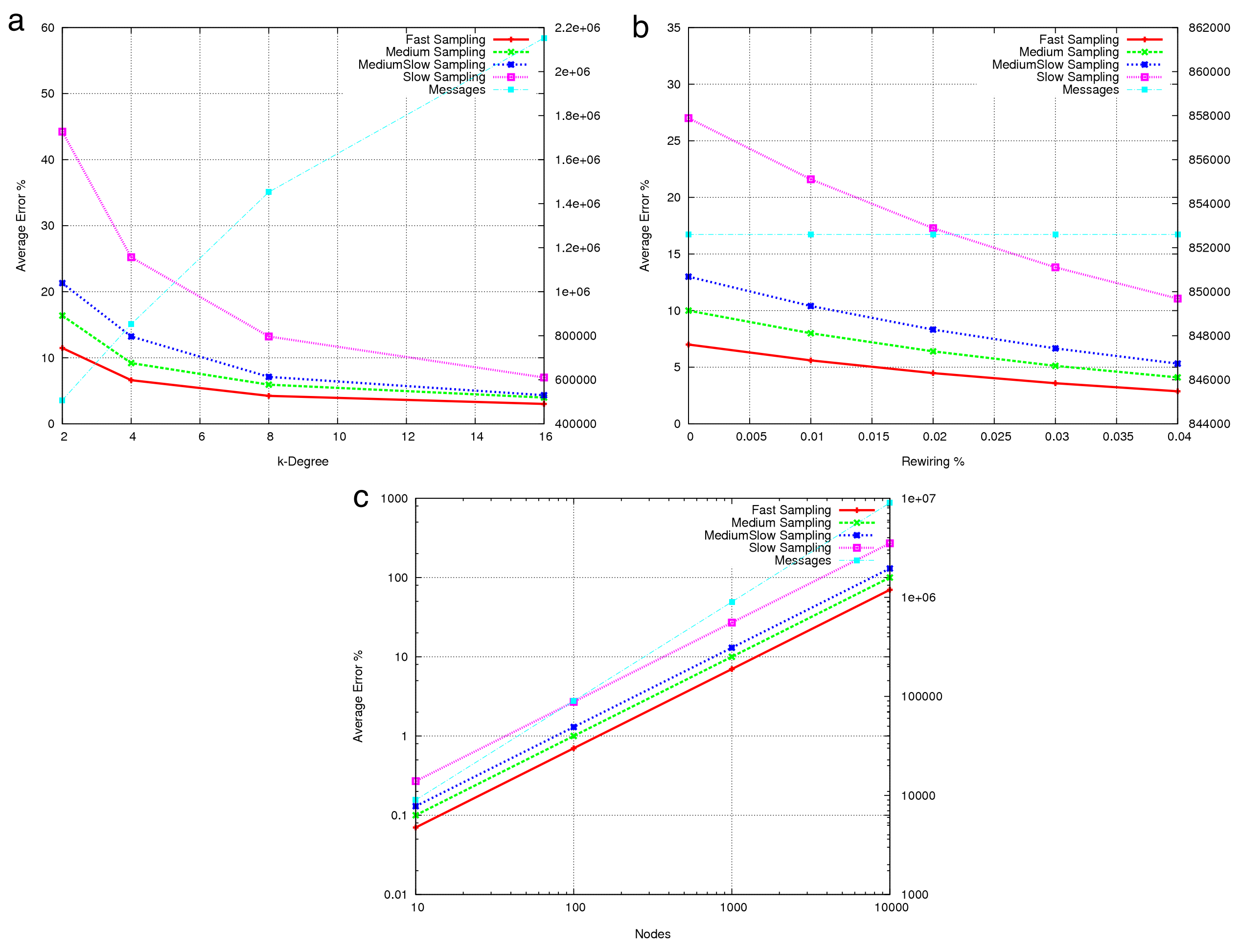

Fig. 11. GEAopt performances varying network (a) k degree, (b) rewiring factor and (c) size.

the network density is the same of doubling ω (fraction of nodes actually involved in the gossiping).

The three (b) graphs ($k = 4, n = 1000, C = 0$) show that—for all the three approaches—the average error decreases quickly by increasing the percent of random links. This outlines the relevance of the network topology in this kind of diffusion schemes. The more the network is random, the more the network diameter decreases, the lower the average errors. That is, although the 3 approaches work both for wireless sensor network and for P2P networks, they are more effective in networks with a low diameter.

The three (c) graphs ($k = 4, \alpha = 0, C = 0$) show the scalability of the algorithms varying the network size from 10 to 10 000 nodes. As expected, for all the three approaches, the average error increases with the number of participating nodes. This happens because the number of iterations needed by the schemes to converge increases with the number of nodes. However, a sublinear trend guarantees good scalability.

Overall, these graphs show that the three approaches exhibit very similar behavior while changing their common operational parameters, when executing on different network structures, and under different dynamics. The analysis becomes interesting when comparing the relative performances of the three approaches.

To compare the data reported in the previous graphs, in Fig. 13 we reported them in a different view. In particular we reported the average error rate of each of the three approaches (GEA, GEAopt, EVP) varying the k, the number of nodes (n), the rewiring percent (α) and the clustering factor (C). For the sake of clarity we decided to report only the data referring to $t = T(medium)$. Also in this graph, the left vertical axis represents the average error, while the right vertical axis represents the number of messages being exchanged.

It is clear from the graphs that GEA is the worst out of the three. Using the same number of messages it produces and average error consistently higher than the other two. EVP is also slightly better that GEAopt. In fact, it produces on average an error 5% lower than GEAopt at no additional communication costs. It is also worth mentioning the effect of the size of the network. With small networks, which have low convergence time, GEAopt outperforms EVP. By increasing the size the network the result changes, and EVP is able to achieve an error consistently lower. In other words, it appears that EVP is more scalable the GEA and GEAopt.

To further test the behavior of the system and to verify that the algorithms exhibited the same overall behavior in the presence of network dynamics, we conducted some experiments comparing the behavior of the 3 approaches upon dynamic changes in the network topology.

We developed experiments in which nodes connect and disconnect from the network (i.e., read *null* and do no longer participate in the gossip when detaching from the network, and gossip again when reconnecting). We simulated that in every experiment—250 iterations long—the 25% of the nodes are churning every epoch. In particular, for every iteration, we randomly connect or disconnect some nodes until the desired percentage has been reached. In order to simulate a realistic network fault, disconnected nodes stop sensing the environment and updating their aggregated estimates until they are connected to the network again.

Fig. 14 illustrates the behavior of the different algorithms under these assumptions. It is possible to see that the impact of the

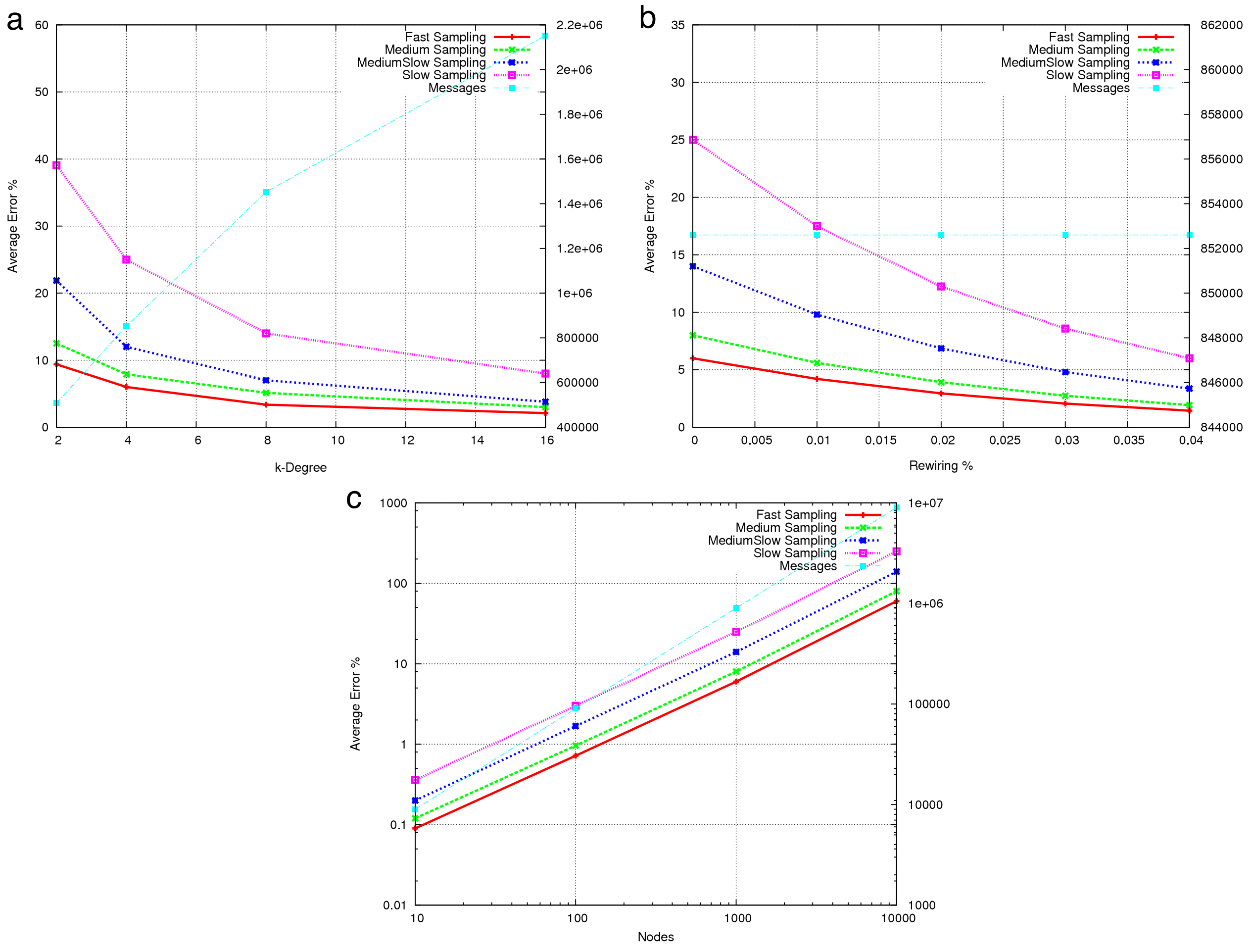

Fig. 12. EVP performances varying network (a) k degree, (b) rewiring factor and (c) size.

examined network and algorithmic parameters remain the same even in presence of network dynamism. Moreover, we experimentally verified that two different forms of dynamisms (i.e., network's and environment's dynamism) have similar impacts on the system.

Our proposal is able to manage the dynamism of the values measured over the network and, in the most of the cases, is more accurate that the other two. In our opinion, it has three key advantages: (i) it has no need of periodic restarts and could be taken into account whenever very large networks are involved; (ii) it can produce average errors slightly lower that epoch-based solutions which are, moreover, less predictable and inevitably characterized by periodic error spikes as shown in Fig. 4; (iii) it has interesting autonomic properties, since it does not require a fine-tuning of its parameters, as depicted in Fig. 8.

6. Related work

Given the size and the dynamism of modern distributed and pervasive computing scenarios (like P2P and wireless sensor networks), several innovative approaches to collect properties about the global state of such systems have been proposed by the research community. In this section, we first overview mechanisms and algorithms to collect data from a distributed system, and emphasize how these approaches deal with the dynamism of the environment and of the underlying network. Then, we discuss more specifically those approaches applying the bio-inspired approach of "evaporation" to distributed systems.

6.1. Data gathering in distributed systems

Most of the works on data gathering and aggregation in distributed systems assume the presence of a fixed number of sinks (i.e., base stations or data centers) to which gathered data should flow. In wireless sensor networks, this typically consists in collecting all the sensed data into a base station for further analysis. In large-scale distributed computing, this translates in the need to collect logs and data reports into a limited set of control centers. This is the case, for example, of network intrusion detection systems (NIDSs) and of those services monitoring the performance of the network. In such situations, the most general and basic approach is that of having the nodes of the network build a spanning-tree rooted at the sink and supporting the routing of data towards it [28]. To deal with the transfer of possibly large amounts of data, several forms of in-network data aggregation (e.g., averaging or max/min determination) can be performed as data climb the tree, with the goal of reducing communications between nodes [29–31].

In some recent scenarios [32,33], wireless sensor networks have moved from being closed special-purpose systems devoted to monitor specific phenomena [33–35] to act as the basis of a truly pervasive and dense shared infrastructure, available for general-purpose usage by a variety of users. Just to make some examples: cars in a city can access sensors around to obtain on-the-fly updated traffic information; tourists can exploit sensors around to discover urban information and activities; software services can exploit the information obtained by local nodes to contextualize

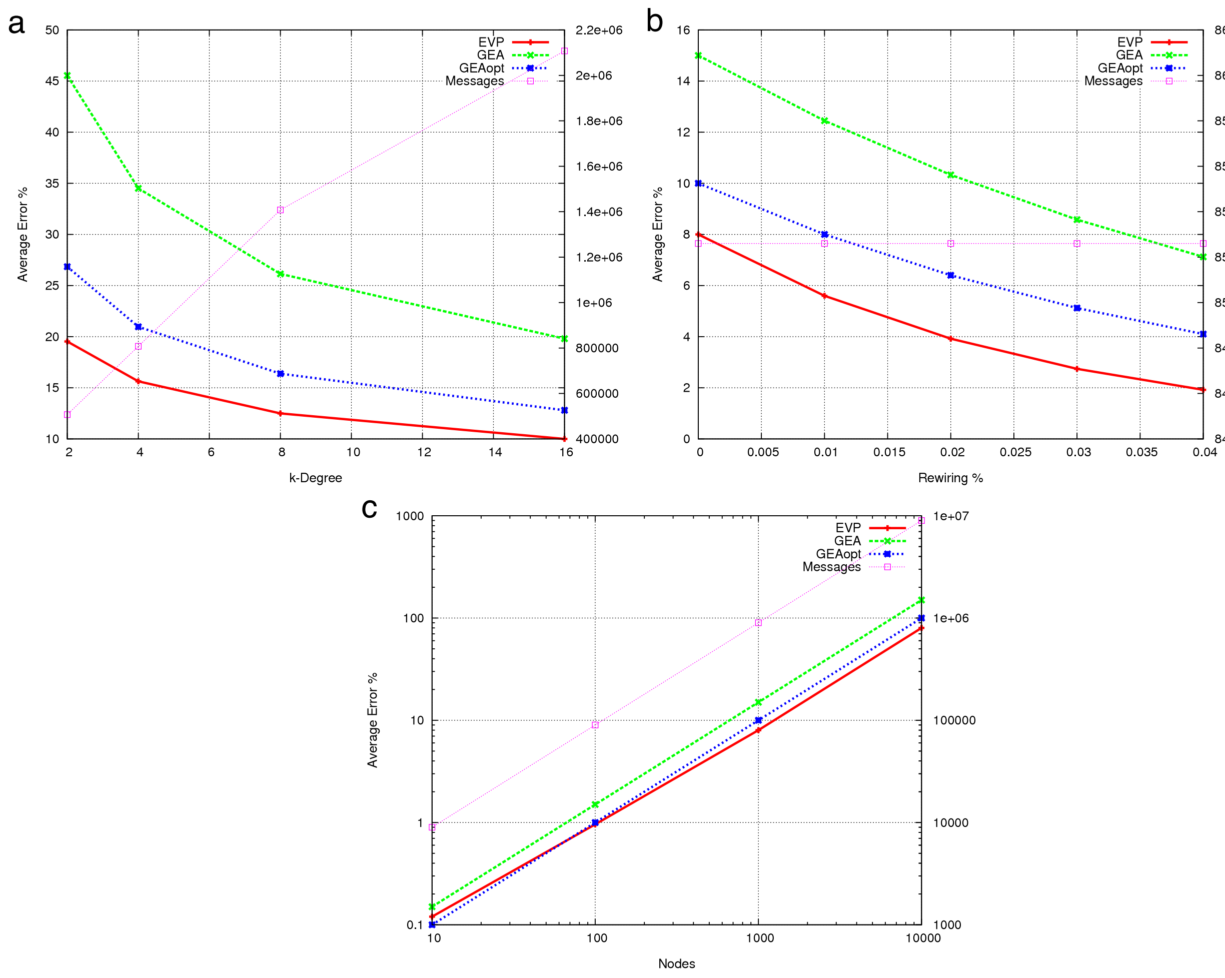

Fig. 13. The average error rate of the three approaches (GEA, GEAopt, EVP) in presence of environmental dynamism. For the sake of uniformity with the previous experiments we investigated: (a) the k-degree ($n = 1000$, $\alpha = 0$, $C = 0$), (b) the rewiring percent of the network α ($k = 4$, $n = 1000$, $C = 0$), (c) the number of nodes of the network n ($k = 4$, $\alpha = 0$, $C = 0$). In these experiments we used a sampling period $t = medium$.

their behavior and improve users' satisfaction. Similarly, in the area of advanced distributed computing and networking, [36], an increasing number of problems requires distributed components (other than centralized services) to have access to data from multiple nodes in order to tune and self-configure themselves.

Clearly, in all the above scenarios, it becomes fundamental to have global information about the status of the network be distributed and made available at the level of individual nodes. However, tree-based approaches can hardly apply as a general solution. In fact, the costs of building a tree on demand for many possible users and components at different and varying locations would be unbearable, both in terms of resource consumption and response times. Some algorithmic optimizations for tree-based approaches specifically conceived for access by mobile users and distributed sinks have been proposed [37,7,38,39]. For instance, [39] proposes exploiting swarm intelligence techniques to dynamically adapt routing trees based on current availability of resources of nodes, thus making it possible to exploit at the best and in a more balanced way the load on the nodes. In any case, neither this one nor other proposed optimizations fully eradicate—but only smooth—the identified flaws of tree-based approaches.

A totally different approach that has been recently proposed, and that shares the key objective of our proposal, relies on pre-aggregating data in the network from within the various nodes of the network itself. Thus, queries by multiple users and services can be answered immediately without the additional burden related to the on-demand building of routing trees. In [40], a cluster-based data collection scheme is proposed in which the network partitions into a set of clusters where all the nodes are at a 1-hop distance from a dynamically elected cluster head. Then, each cluster head can easily pre-acquire a global view of the sensorial situations in its cluster and, if queried, it can provide with a quick aggregated sketch of the situation around it. In [41], a more elaborated approach is proposed in which each node in the network can compute, at predefined communication/resource consumption costs, aggregated information about information in its neighborhood, also making it possible to concurrently compute aggregation functions for neighborhoods of different (exponentially enlarging in terms of network hops) sizes. However, this proposals are conceived for very specific scenarios, while our proposal aims at being a general-purpose on.

As a final note, several research works in the area of middleware and programming languages for distributed systems start recognizing the need to support direct access to sensor data by multiple users and services. These works mostly focus on defining suitable general-purpose primitives and language constructs (together with the supporting middleware infrastructures) to enable users and services to flexibly query the network and obtain information about individual data and aggregated data related to specific network sub-graphs. Examples of these approaches include Region Streams [42], Logical Neighborhood [43], or TeenyLIME [44]. These kinds of research are somewhat complementary to our proposal. The algorithm we propose could in fact

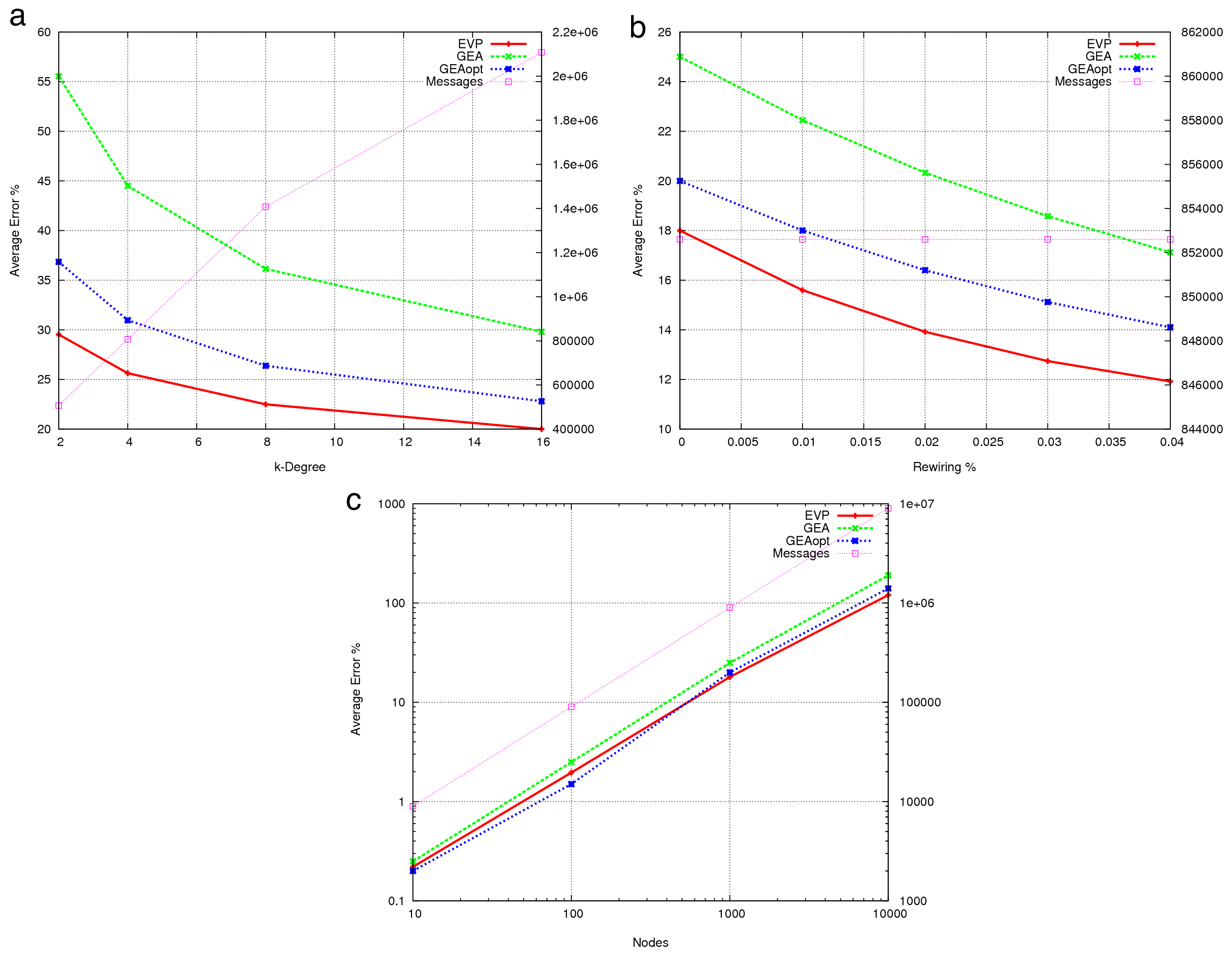

Fig. 14. The average error rate of the three approaches (GEA, GEAopt, EVP) in presence of network churn. For the sake of uniformity with the previous experiments we investigated: (a) the k-degree ($n = 1000, \alpha = 0, C = 0$), (b) the rewiring percent of the network α ($k = 4, n = 1000, C = 0$), (c) the number of nodes of the network n ($k = 4, \alpha = 0, C = 0$). In these experiments we used a sampling period $t = medium$.

serve as the basis for further enriching the expressiveness and the power of such approaches.

6.2. Evaporation-based approaches

Inspired by early seminal works on ant-based routing [23], having shown that evaporation of pheromone trails (i.e., of routing paths) is an essential ingredient to keep pheromone-based routing tables updated despite network dynamism and traffic conditions, other works have adopted the idea of evaporation in distributed systems.

Some recent works exploit the bio-inspired idea of *evaporation* to effectively deal with the problems of network and information dynamism. Swarm Linda [45] is one of the first examples of middleware adopting this concept. Swarm Linda consists of a distributed tuple space adopting bio-inspired mechanisms to deal with system-level issues. In particular, one thorny problem of tuple spaces regards garbage-collecting tuples once they are not useful any more. Swarm Linda deals with this problem with the idea of tuple *fading* (i.e., evaporation). Tuples contain a numeric value that is slowly reduced by the system, and it is restored to its original value once the tuple is accessed. Once the value reaches zero (since the tuple has not been accessed for long time), the tuple is deleted from the system.

In [46], evaporation is used to deal with information ageing in knowledge networks (i.e., distributed information system). This work presents a framework to organize and structure context information to be used in pervasive computing scenarios. Different pieces of information (called knowledge atoms) are combined together to infer novel data that is reified by means of novel (dynamically generated) atoms. The number of new information that can be derived from an existing knowledge base is huge and requires some mechanisms to be contained. A sort of *evaporation*-based mechanisms has been enforced to contain the proliferation of information. Information are aged and then deleted by gradually decreasing a counter value indicating when it was last accessed.

Chemical TuCSoN [47] is a middleware supporting the developing of self-organizing applications. The middleware is built around the idea of tuple spaces that are programmable in that they allow to install rules automatically triggered by the insertions and removals of tuples. The idea of chemical TuCSoN is to have chemical laws encoded by means of the tuple space's rules and use the tuples like chemical elements. This kind of middleware support different kind of self-organization patterns useful for many applications. In particular the *Decay* primitive used in this system realizes the evaporation of tuples from the space.

Similarly, TOTA [48] is a middleware supporting self-organizing applications by means of field-based data structures (i.e., gradients) spread across the network. TOTA supports different kind of such field-based data structures and in particular it supports time-dependent data structures that change their value over time. In many applications these time-dependent structures enforce an evaporation kind of mechanisms to slowly delete the data if not constantly reinforced.

Despite the existence of the above possible exploitation of evaporation, our specific proposal for applying evaporation in diffusive aggregation is, to the best of our knowledge, new.

7. Conclusions and future works

In this paper, we have presented an innovative bio-inspired approach to handle dynamics in diffusive aggregation schemes, relying on the evaporation of aggregated values, and have compared it with more traditional epoch-based approaches based on periodic restarts. Our proposal is able to effectively manage dynamics of the values measured over the network, can apply to both wireless sensor networks and P2P networks and, in the most of the cases, is more accurate that epoch-based approaches. In particular, experimental results show that the proposed evaporative solution is to be preferred in very large networks, in networks with high sampling rates, and in those particular setups where periodic restarts should be avoided. Yet, for small-size networks, epoch-based solutions work pretty well too.

There are a number of open issues and future research that could further improve our work:

- It would be interesting to realize mechanisms to automatically tune the evaporation rate to the specific environmental conditions. This process requires sensors to monitor the environment dynamics and to set the evaporation rate autonomously in order to optimize the graph in Fig. 8. This could lead different evaporation rates being set in different parts of the network.
- It would be also necessary to investigate how the evaporation mechanism can behave in the presence of different network topologies and larger networks than we have tested so far. Accordingly, we will try to develop larger-scale and more realistic simulations of the distributed systems setting.
- It would also be interesting to evaluate if it could be valuable to combine different mechanisms together (e.g., epoch-based approaches together with evaporation) and being able to autonomously switch between them to globally optimize the behavior of the system.
- Also with regard to the gossiping itself, it would be interesting to create a system combining gossiping and routing trees and to switch between them dynamically on the basis of the current network conditions.

In conclusion, and despite the good results achieved so far, several open research directions can be taken to further improve and assess it.

References

[1] D. Estrin, D. Culler, K. Pister, G. Sukjatme, Connecting the physical world with pervasive networks, IEEE Pervasive Computing 1 (1) (2002) 59–69.
[2] J. Crowcroft, Toward a network architecture that does everything, Communication of the ACM 51 (1) (2008) 74–77.
[3] B. Hayes, Cloud computing, Communication of the ACM 51 (7) (2008) 9–11.
[4] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies, ACM Computing Surveys 36 (4) (2004) 335–371.
[5] I. Rodero, F. Guim, J. Corbalan, L. Fong, M. Sadjadi, Grid broker selection strategies using aggregated resource information, Future Generation Computer Systems 26 (1) (2010) 72–86.
[6] S. Nath, P. Gibbons, Synopsis diffusion for robust aggregation in sensor networks. Technical report, Intel Research, 2003.
[7] T. Schoellhammer, B. Greenstein, D. Estrin, Hyper: A routing protocol to support mobile users of sensor networks. Technical report, Center for Embedded Network Sensing, 2006.
[8] P. Eugster, R. Guerraoui, A.M. Kermarrec, L. Massoulieacute, Epidemic information dissemination in distributed systems, Computer 37 (5) (2004) 60–67.
[9] M. Jelasity, A. Montresor, O. Babaoglu, Gossip-based aggregation in large dynamic networks, ACM Transactions on Computer Systems 23 (3) (2005) 219–252.
[10] O. Babaoglu, G. Canright, A. Deutsch, G.D. Caro, F. Ducatelle, L. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, T. Urnes, Design patterns from biology for distributed computing, ACM Transactions on Autonomous and Adaptive Systems 1 (1) (2006) 26–66.
[11] M. Jelasity, A.M. Kermarrec, Ordered slicing of very large-scale overlay networks, in: IEEE International Conference on Peer-to-Peer Computing, Cambridge, UK, 2006, pp. 117–124.
[12] N. Bicocchi, M. Mamei, F. Zambonelli, Self-organizing spatial regions for sensor network infrastructures, in: IEEE International Conference on Advanced Information Networking and Applications, vol. 2, IEEE Computer Society, Los Alamitos, CA, USA, 2007, pp. 66–71.
[13] A. Corradi, L. Leonardi, F. Zambonelli, On the effectiveness of different diffusive load balancing policies in dynamic applications, in: International Conference and Exhibition, HPCN Europe, Amsterdam, NL, 1998, pp. 274–283.
[14] M. Brunato, R. Battiti, A. Montresor, Gosh! gossiping optimization search heuristics, in: International Conference on Learning and Intelligent Optimization, 2007.
[15] A. Dimakis, A. Sarwate, M. Wainwright, Geographic gossip: Efficient aggregation for sensor networks, in: International Conference on Information Processing in Sensor Networks, Nashville, TN, USA, 2006.
[16] S. Voulgaris, M. van Steen, K. Iwanicki, Proactive gossip-based management of semantic overlay networks, Concurrency and Computation: Practice and Experience 19 (17) (2007) 2299–2311.
[17] P. Costa, V. Gramoli, M. Jelasity, G.P. Jesi, E.L. Merrer, A. Montresor, L. Querzoni, Exploring the interdisciplinary connections of gossip-based systems, Operating Systems Review—Special Topic: Gossip-Based Networking 41 (4) (2007) 51–60.
[18] A. Montresor, Intelligent gossip, in: International Symposium on Intelligent Distributed Computing, Catania, IT, Invited paper, 2008.
[19] P. Skraba, Q. Fang, A.T. Nguyen, L.J. Guibas, Sweeps over wireless sensor networks, in: International Conference on Information Processing in Sensor Networks, Nashville, TN, USA, 2006, pp. 143–151.
[20] A. Forestiero, C. Mastroianni, G. Spezzano, Reorganization and discovery of grid information with epidemic tuning, Future Generation Computer Systems 24 (8) (2008) 788–797.
[21] M. Mamei, R. Menezes, R. Tolksdorf, F. Zambonelli, Case studies for self-organization in computer science, Journal of Systems Architecture 52 (8–9) (2006) 443–460.
[22] J. Hong, S. Lu, D. Chen, J. Cao, Towards Bio-Inspired Self-Organization in Sensor Networks: Applying the Ant Colony Algorithm, IEEE Computer Society, Washington, DC, USA, 2008, pp. 1054–1061.
[23] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence. From Natural to Artificial Systems, Oxford University Press, Oxford, UK, 1999.
[24] D. Watts, S. Strogatz, Collective dynamics of small-world networks, Nature 393 (1998) 440–442.
[25] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: SIGCOMM Conference, ACM Press, San Deigo, CA, USA, 2001.
[26] M. Ripeani, A. Iamnitchi, I. Foster, Mapping the gnutella network, IEEE Internet Computing 6 (1) (2002) 50–57.
[27] A. Barabasi, E. Bonabeau, Scale-free networks, Scientific American 288 (2003) 60–69.
[28] J. Polastre, R. Szewcyk, A. Mainwaring, D. Culler, J. Anderson, Analysis of wireless sensor networks for habitat monitoring, Wireless Sensor Networks (2004) 399–423.
[29] S. Madden, J. Hellerstein, Distributing queries over low-power wireless sensor networks, in: ACM International Conference on Management of Data, Madison, WI, USA, 2002.
[30] J. Gehrke, S. Madden, Query processing in sensor networks, IEEE Pervasive Computing 3 (1) (2004) 46–65.
[31] A. Boulis, S. Ganerival, M. Srivastava, Aggregation in sensor network: An energy-accuracy trade-off, in: International Workshop on Sensor and Actuator Network Protocols and Applications, Anchorage, AK, USA, 2003.
[32] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, J. Reich, Mobiscopes for human spaces, IEEE Pervasive Computing 6 (2) (2007) 20–29.
[33] O. Riva, C. Borcea, The urbanet revolution: Sensor power to the people!, IEEE Pervasive Computing 6 (2) (2007) 41–49.
[34] N. Ramanathan, L. Balzano, D. Estrin, M. Hansen, T. Harmon, J. Jay, W. Kaiser, G. Sukhatme, Designing wireless sensor networks as a shared resource for sustainable development, in: International Conference on Information and Communication Technologies and Development, Berkeley, CA, USA, 2006.
[35] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, R. Nagpal, Firefly-inspired sensor network synchronicity with realistic radio effects, in: ACM International Conference on Embedded Networked Sensor Systems, San Diego, CA, USA, 2005.
[36] S. Dobson, S. Denazis, A. Fernández, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, A survey of autonomic communications, ACM Transactions on Autonomous and Adaptive Systems 1 (2) (2006) 223–259.
[37] G. Cormode, M. Garofalakis, S. Muthukrishnan, R. Rastogi, Holistic aggregates in a networked world: Distributed tracking of approximate quantiles, in: ACM International Conference on Management of Data, Baltimore, MD, USA, 2005.
[38] S. Kabaday, C. Julien, Scenes: Abstracting interaction in immersive sensor networks, Journal on Pervasive and Mobile Computing 3 (6) (2007) 635–658.
[39] H. Yang, F. Ye, B. Sikdar, A swarm intelligence based protocol for data acquisition in networks with mobile sinks, IEEE Transactions on Mobile Computing 7 (8) (2008) 931–945.

[40] M. Lotfinezhad, B. Liang, E. Sousa, Adaptive cluster-based data collection in sensor networks with direct sink access, IEEE Transactions on Mobile Computing 7 (7) (2008) 884–897.
[41] R. Sakar, X. Zhu, J. Gao, Hierarchical spatial gossip for multi-resolution representations in sensor networks, in: International Conference on Information Processing in Sensor Networks, Cambridge, MA, USA, 2007.
[42] R. Newton, M. Welsh, Region streams: Functional macroprogramming for sensor networks, in: International Workshop on Data Management for Sensor Networks, Toronto, Canada, 2004.
[43] G. Mottola, G.P. Picco, Logical neighborhoods: A programming abstraction for wireless sensor networks, in: IEEE International Conference on Distributed Computing in Sensor Systems, San Francisco, CA, USA, 2006.
[44] P. Costa, L. Mottola, A. Murphy, P. Picco, Programming wireless sensor networks with the teenylime middleware, in: ACM Middleware Conference, Newport Beach, CA, USA, 2007.
[45] R. Tolksdorf, R. Menezes, Using swarm intelligence in linda systems, in: Engineering Societies in the Agents World, Springer Verlag, 2004.
[46] M. Casadei, S. Montagna, M. Viroli, F. Zambonelli, A biochemical metaphor for developing eternally adaptive service ecosystems, in: ACM Symposium on Applied Computing, Hawaii, HI, USA, 2009.
[47] M. Viroli, M. Casadei, Biochemical tuple spaces for self-organising coordination, in: COORDINATION'09: Proceedings of the 11th International Conference on Coordination Models and Languages, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 143–162.
[48] M. Mamei, F. Zambonelli, Programming pervasive and mobile computing applications: The total approach, ACM Transaction on Software Engineering and Methodology 19 (4) (2009).

Nicola Bicocchi is a postdoctoral fellow in Computer Science at the University of Modena and Reggio Emilia. He received the Laurea degree in Computer Engineering from the University of Modena and Reggio Emilia in 2004, discussing a thesis on "Techniques for Content-based video adaptation". He received the Ph.D. in Computer Science from the same University in 2009, with a thesis on "Self-Organizing Data Ecologies for Pervasive Computing Scenarios". He has been visiting researcher at University of Ulster and University of New Brunswick (Canada).

His current research interests include pervasive computing, affective computing, data mining and analysis for behavioural prediction and network security. In these areas, he has published about 20 papers in international journals and conferences. He participated, during his Ph.D. in the European Integrated Project "CASCADAS" and, during his year in Canada, in a project founded by Defense Canada.

Marco Mamei is assistant professor in Computer Science at the University of Modena and Reggio Emilia, since September 2004. He received the Laurea degree in Computer Engineering from the University of Modena and Reggio Emilia, in February 2001, discussing a thesis on "Web Technologies for Mobile Computing" developed at Telecom Italia Labs. He received the Ph.D. in Computer Science from the same University in April 2004, with a thesis on "Field-based Coordination in Dynamic Pervasive Networks". He has been visiting researcher at Telecom Italia Lab (IT), Nokia Research Center (USA), Harvard University (USA), Cycorp Europe (SLO) and Yahoo! Research (SP). His current research interests include: applications and infrastructures for pervasive and mobile computing, data mining for location-aware computing. In these areas, he has published over 70 papers in international fora and 1 book (monograph), received several awards, and has been invited speaker and tutorialist in some international conferences and workshops. He participated in some European and Italian project. In particular, the European Integrated Project "CASCADAS", and the project "Infrastructures for Mobile Ad-Hoc Networks" funded by the Italian Research Ministry. He is member of the program committee and co-organizer of several major conferences and workshops.

Franco Zambonelli is professor in Computer Science at the University of Modena and Reggio Emilia since 2001. He obtained the Laurea degree in Electronic Engineering in 1992, and the Ph.D. in Computer Science in 1997, both from the University of Bologna. His current research interests include: distributed and pervasive computing, autonomic computing and communication, software engineering for large-scale agent systems. In these areas, he has published 1 monograph, over 40 papers in international peer-reviewed journals, co-edited 7 books, and received several best paper awards. He has been invited speaker and tutorialist in several international conferences and workshops. He is in the Editorial Board several major journals, there included the Elsevier Journal of Pervasive and Mobile Computing and of the ACM Transactions on Autonomous and Adaptive Systems. He is member of the program committee of several major conferences and workshops in the areas of distributed, pervasive, and autonomic computing systems. He has been member of the management committee of the European Network of Excellence ≪ Agentlink II≫ (2000–2003) and, within the same network, coordinator of the Special Interest group on ≪Methodologies and Software Engineering for Agent Systems≫. He has been Scientific Manager for the European Integrated Project "CASCADAS", local unit coordinator for the project "Infrastructures for Mobile Ad-Hoc Networks" funded by the Italian Research Ministry, and for the project "Ambient Intelligence for a Friendly City" funded by Regione Emilia Romagna.