Self-organizing Virtual Macro Sensors

NICOLA BICOCCHI, MARCO MAMEI, FRANCO ZAMBONELLI
University of Modena and Reggio Emilia

The future large-scale deployment of pervasive sensor network infrastructures calls for mechanisms enabling the extraction of general-purpose data at limited energy costs. The approach presented in this paper relies on a simple algorithm to let a sensor network self-organize a virtual partitioning in correspondence of spatial regions characterized by similar sensing patterns, and to let distributed aggregation of sensorial data take place on a per-region basis. The result of this process is that a sensor network can be modeled as a collection of virtual macro sensors, each associated to a well-characterized region of the physical environment. Within each region, each physical sensor has the local availability of aggregated data about its region and is able to act as an access point to such data. This feature promises to be very suitable for a number of emerging usage scenarios. Our approach is described, evaluated in both a simulation environment and a real test bed, and quantitatively compared with related works in the area. Current limitations and areas of future development are also discussed.

Categories and Subject Descriptors: D.2.7 [**Software Engineering**]: Distribution and Maintenance; H.4.0 [**Information Systems Applications**]: General; C.2.4 [**Computer-Communication Systems**]: Distributed Systems

Additional Key Words and Phrases: Pervasive Computing, Self-Organization, Sensor Networks

1. INTRODUCTION

In the next few years, the deployment of sensor network systems [Abdelzaher et al. 2007; Want and Pering 2005] is likely to increase. Most likely, such large scale deployment will induce a radical change in their *Raison d'Étre*. Rather than being special-purpose systems devoted to monitor specific phenomena [Riva and Borcea 2007; Ramanathan et al. 2006; Werner-Allen et al. 2005], as they are today, they will form the basis of a truly pervasive and dense shared infrastructure, available for general-purpose usage by a variety of users. This trend can already be observed through several examples: cars in cities can access nearby sensors to obtain on-the-fly updated traffic information; tourists can exploit sensors around to discover urban information and activities; in the case of a disaster, sensors can support robots and humans during emergency operations; software services can exploit the information obtained by local sensors to contextualize their behavior and improve users' satisfaction. In addition, this trend is even more evident when considering social sensors' ideas: users reporting information about events happening in their environment via social network application running on their mobile terminals (e.g., via Twitter posts) [Sakaki et al. 2010].

This change in the nature of sensor networks will also change the ways by which such systems are accessed and exploited (see Figure 1). As of today, most sensor network systems are conceived to report data to predefined base stations by routing data through the network to a fixed sink [Polastre et al. 2004; Boulis et al. 2003]. Clearly, such fixed sinks will be present also in future sensor network scenarios. However:

—sinks will have to collect general-purpose data for remote users that want to discover about various events/phenomena happening somewhere in the world [Baumgarten et al. 2006; Castelli et al. 2007];

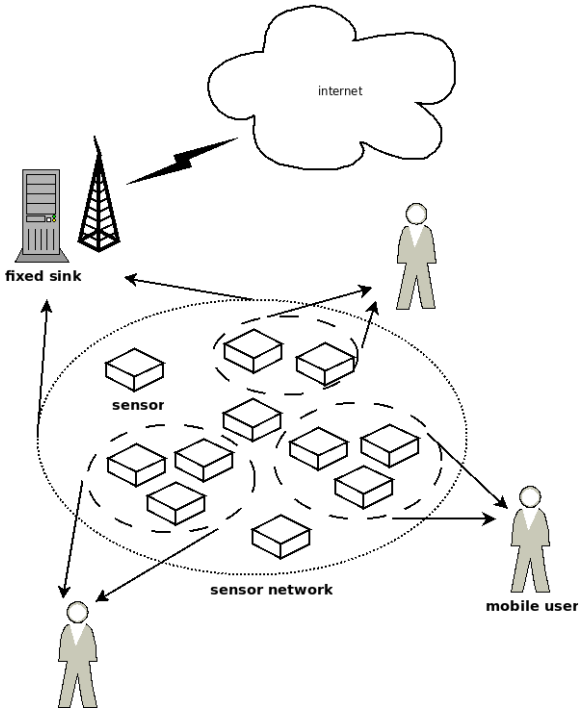


Fig. 1. Shared sensor network infrastructure.

—sensors will have to report data on-demand to multiple and mobile users that can exploit short-range wireless links to access local sensors around and retrieve users-specific data [Muller and Alonso 2005; Curino et al. 2005; Costa et al. 2007; Lu et al. 2005];

This novel perspective introduces peculiar requirements. First, it is expected that the sensor network, despite being intensively used in unpredictable ways, will be able to control its energy consumption. In other words, the energy costs will have to be bounded and balanced over the network, so as to ensure a minimal guaranteed lifetime or, for self-rechargeable devices [Jiang et al. 2005], a predefined average power consumption. Second, it is expected that the network will be able to provide users (whether remote or local) with expressive and compact information related to the phenomena under sensing rather than raw individual sensor data. In fact, in large sensor networks generating huge amounts of data, dealing with the transfer and the ex-post analysis of individual sensor data can be unmanageable. Finally, the network should quickly answer users' requests. Since users can be highly mobile (e.g., a car) a late answer to a query can either fail to reach the user or reach him at a location where the answered information is useless.

These requirements poses to traditional tree-based, *reactive* approaches (discussed in section 2) tremendous challenges. The main contribution of this paper is to propose an innovative approach to tackle them. The idea underlying our proposal is that of delegating to the sensor network the execution of distributed algorithms that – by *proactively* running over the network with predictable energy costs – can enforce:

—The adaptable partitioning of the network into spatial regions characterized by similar

data patterns, via a self-organizing overlay network;

—The distributed aggregation of whatever sensorial data on a per-region basis;

The result of this process is that a sensor network can be modeled as made up of *virtual macro sensors*, each one associated to a well-characterized region of the physical environment (i.e., a region exhibiting a uniform pattern for some specific property such as a light, temperature, etc.). Within each region, each physical sensor has the local availability of aggregated data of its region and might act as an access point to such data.

This approach allow multiple and mobile users to promptly access global information about the surrounding environment by simply querying the closest sensor, without any additional costs for the sensor network. Also, it makes possible to effectively transfer aggregated data towards a centralized collection point in a more compact and efficient way, yet avoiding the loss of information typical of global aggregation algorithms. Furthermore, this process is independent from the actual density, topology, and dynamics of the underlying physical sensor network.

Performance studies performed both in a simulation environment and on a real test-bed confirm the effectiveness of our proposal and its potential for being exploited in a wide range of applications scenarios.

The remainder of this paper is organized as follows. Section 2 motivates our proposal and discusses related work in the area. Section 3 illustrates the concept of virtual macro sensors, describes the algorithms supporting the abstraction and the solutions to adaptively deal with network and environmental dynamics. Section 4 discusses experimental results. Section 5 compares the virtual macro sensor approach with others in the area. Section 6 discusses potential applications and current limitations. Section 7 concludes the paper.

2. MOTIVATIONS AND RELATED WORK

Most of the works on data gathering and aggregation in sensor networks assume the presence of fixed sinks (i.e., base stations) to which sensed data should flow. In such situations, the basic approach is that of having sensors build a tree rooted at the sink and supporting the routing of sensed data towards it [Polastre et al. 2004]. 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 from sensors climb the tree [Madden and Hellerstein 2002; Gehrke and Madden 2004; Boulis et al. 2003], with the goal of reducing communications between sensors and, thus, the energy costs. However, such aggregation algorithms does not generally account for the data patterns exhibited by phenomena under sensing, and are thus at risk of being either ineffective or very lossy.

As far as multiple and mobile users are concerned, tree-based approaches can hardly apply as a general solution. In fact, the costs of building a tree on demand for many possible users at different and varying locations would be unbearable, both in terms of energy costs (proportional to the number of queries and to the spatial extent of such queries), energy unbalances (if queries do not evenly distribute over the network) and response times. Some algorithmic optimizations for tree-based approaches specifically conceived for access by mobile users have been proposed [Cormode et al. 2005; Schoellhammer et al. 2006; Kabaday and Julien 2007; Yang et al. 2008]. For instance, [Yang et al. 2008] proposes exploiting swarm intelligence techniques to dynamically adapt routing trees based on current energy conditions of nodes, thus making it possible to exploit at the best and in a more balanced way the energy of sensor nodes. In any case, neither this one nor other proposed optimiza-

tions fully eradicate the identified flaws of tree-based approaches.

A different approach is presented in [Khelil et al. 2009]. The proposed mechanisms (gMAP) opportunistically exploits node mobility to collect data of interest. While node mobility is a reasonable assumptions in some settings, in other scenarios it might be very difficult and costly to implement such an approach. Our proposal does not take advantage of nodes' mobility, but possible integration between this and our mechanism could be useful.

A totally different approach that has been recently proposed, and that shares one of the key characteristics of our proposal, relies on somewhat pre-aggregating data in the network (or in regions of the network, which is a more natural choice for large-scale sensor networks). Thus, queries by multiple and mobile users can be answered immediately without the additional burden related to the on-demand building of routing trees. In [Lotfinezhad et al. 2008], 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, provide users with a quick aggregated sketch of the situation around it. In [Sarkar et al. 2007], a more elaborated approach is proposed in which each node in the sensor network can compute, at pre-defined communication/energy costs, aggregated information about sensorial 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. Both these approaches share with our proposal the possibility of making available, locally at the nodes of the network at each node and at pre-defined costs, pre-aggregated information about the situation in the neighborhood. However, the definition of the regions on which to produce aggregated data is mostly network driven, and does not take into account the specific characteristics of the environment being monitored or of the phenomena being sensed (as our proposal does). This may cause globally-aggregated properties to lose relevant information and significance.

The issue of recognizing regions of a sensor network characterized by similar properties of sensed data is considered very important to improve the reliability and capability of monitoring, and to acquire a higher-level perspective of the phenomena under sensing. Indeed, some in-network algorithms for self-organization of region partitioning in sensor networks have been proposed [Catterall et al. 2002; Panangadan and Sukhatme 2005; Younis et al. 2005], sharing some basic principles with our approach. The key differences with it are that:

—these algorithms require a priori information (i.e., a classification substrate) about the typical patterns exhibited by the environment, while our approach does not require any a priori information and it is fully self-organizing;

—the algorithms are not conceived for other goals than recognizing regions, while our approach goes further, by exploiting regions as a basis for aggregation and for making regions act as sorts of virtual macro sensors. Similar considerations can be made for those algorithms devoted to capture in a distributed way global topological properties of sensor networks and of the phenomena being sensed within them (e.g., contour tracking [Zhu et al. 2008]).

As far as distributed data aggregation is concerned, diffusive algorithms [Corradi et al. 1999] and gossip-based aggregation algorithms [Jelasity et al. 2005; Dimakis et al. 2006;

Skraba et al. 2006] have been proposed as simple yet very effective approaches to compute and make available at each node aggregated values related to some global property of a network. In our approach, we borrow from them by exploiting aggregation algorithms that have a mixed diffusive-based and gossip-based inspiration. However, other than for computing global network values, we use them for computing regional aggregated values (as the basic sensing mechanism of virtual macro sensors), and also propose an innovative solution to deal with network and data dynamics during aggregation. We emphasize that gossip-based algorithm have been also exploited as the basis for partitioning a network into clusters of nodes characterized by similar properties [Jelasity and Kermarrec 2006], a problem similar to the one being addressed in this paper, though with totally different motivations, goals, and scenarios (i.e., large and non-spatially-situated P2P networks instead of sensor networks).

Several research works in the area of middleware and programming languages for sensor networks start recognizing the need to support direct access to sensor data by multiple and mobile users. These works mostly focuses on defining suitable general-purpose primitives and language constructs (together with the supporting middleware infrastructures) to enable users to flexibly query the network and obtain information about individual sensor data and aggregated data related to specific spatial regions. Examples of these approaches include Region Streams [Newton and Welsh 2004], Logical Neighborhood [Mottola and Picco 2006], or TeenyLIME [Costa et al. 2007]. These kinds of researches are somewhat complimentary to our proposal. The algorithms we propose could serve as the basis for further enriching the expressiveness and the power of such approaches.

As a final note, it is worth noting that some works in the complex systems and cellular automata community presents simple and efficient mechanisms for region formation, clustering, and information diffusion. Such kind of work could possibly integrate with our proposal to create more efficient services [Bandini et al. 2001].

3. VIRTUAL MACRO SENSORS

The virtual macro sensor approach (from now on ViMS) builds on the above-described research works and synthesizes them into an innovative proposal suitable for the envisioned emerging scenarios. In a nutshell, the ViMS approach include (see Figure 2):

—A self-organized region formation algorithm, to logically split a sensor network (Figure 2-a) into a set of spatial regions, each characterized by specific environmental patterns in the sensed data (Figure 2-b).

—Localized (i.e., regional) in-network aggregation algorithms to provide each sensor in a region with aggregated information about the overall state of the region (Figure 2-c).

—Peculiar innovative solutions to self-adapt to transitory and dynamic situations. Such solutions ensure that both regions and aggregated information within regions always reflect the current situation of the network and of the environment (Figure 2-d).

This process, which induces predictable and controllable energy costs on the network, enables a sensor network to be perceived and exploited as if it were made up of virtual macro sensors, each one associated to a well-characterized region of the physical environment. Within each region, each physical sensor has the local availability of aggregated data related to its region and can act as an access point to such data. Overall, the ViMS approach is very robust and scalable (as from section 4).

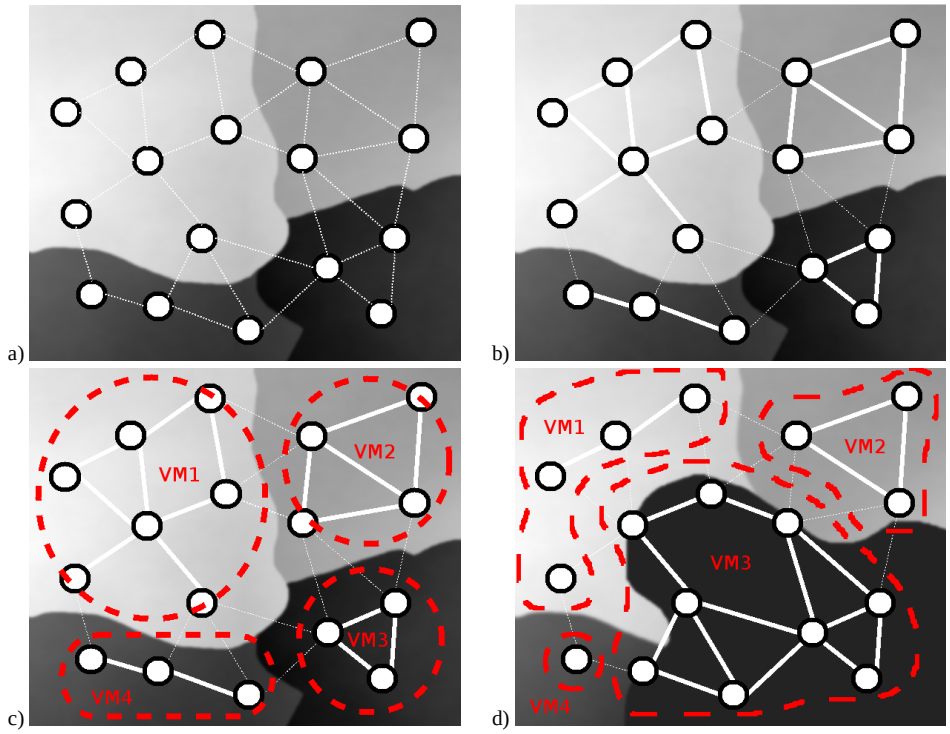


Fig. 2. The ViMS approach in a nutshell: a) a sensor network immersed in an environment characterized by different patterns of some sensed value; b) an overlay of virtual connections self-organize according to such patterns; c) the resulting clusters of sensors associated to each region can act as sorts of virtual macro sensors to provide aggregated information about the data sensed in that region; d) regions and, thus, virtual macro sensors, adaptively re-shape upon changed conditions.

In the following of this Section we go into details about the region formation algorithm, the aggregation algorithms, and the management of transitory and dynamic situations. Also, we shortly discuss the issue of defining proper programming interfaces (APIs) to configure and exploit virtual macro sensors.

3.1 Region Formation

Let us consider a sensor network deployed in an environment in which the value v of some specific environmental property is locally sensed by sensors. The value v could represent a temperature, a light level, or whatever property (even composite) a sensor is able to detect.

The proposed region formation algorithm has the goal of letting sensors self-organize into disjoint sets of spatial regions each characterized by "similar" measures of the property v. Clearly, the underlying assumption is that the environment under monitoring exhibits some spatial discontinuity in its property v, reflecting different environmental regions. Yet, as a matter of fact, both natural landscapes and human-shaped environments exhibit, in most of the cases, such discontinuities.

Organization in regions occurs via a process of building an overlay of virtual weighted links between neighbor nodes (i.e., between nodes which are within the wireless radio range). In particular, nodes belonging to the same region will have strong virtual links,

while neighbor nodes belonging to different regions will have weak (or null) links. As examples: measuring the light level could be used for a sensor network in a building to self-partition on a "per-room" basis (different rooms being characterized by different light levels, while the light level inside a room can be quite homogeneous); measuring the vibration level on a mountain slope could lead to self-organizing a sensor-network into regions associated to surfaces with different geological properties. In general, the region organization can reflect some property of the physical space and can lead to a "logical" organization of sensors, thus making a region to be elected as the spatial extent of a virtual macro sensor.

3.1.1 *Logical Links.* Let p and q be two neighbor sensors, i.e., two sensors whose physical spatial distance is smaller than their wireless radio range r. Let v_p and v_q be the values of a property v sensed by p and q, respectively. Let us assume that a distance function D can be defined for couples of v values. Region formation is then based on iteratively computing the value of a logical link $l(p,q) \in [0,1]$ for each and every node of the system, as in the following $UpdateLink$ procedure:

UpdateLink
if $(D(v_p, v_q) < T)$ {
    $l(p,q) = min(l(p,q) + \Delta, 1)$;
else {
    $l(p,q) = max(l(p,q) - \Delta, 0)$;
}

Where: T is a threshold that determines whether the measured values are close enough for $l(p,q)$ to be re-enforced or, otherwise, weakened; and Δ is a value affecting the reactivity of the algorithm in updating link (more details on all these parameters follow later on).

Based on the above algorithm, it is rather clear that if $D(v_p, v_q)$ is lower than threshold T, $l(p,q)$ will eventually converge to 1 (independently of its initial value). Otherwise it will move towards 0. In general, one could consider two nodes p and q to be in the same region (or *connected*) when $l(p,q)$ is equal or greater than a connection threshold $T_{CONN}, i.e., l(p,q) \geq T_{CONN}$. Here, without loss of generality, we can assume $T_{CONN} = 1$. Transitively, two nodes h and k are defined in the same region if and only if there is a chain of nodes such that each pair of neighbors in the chain are connected with each other.

To improve stability in the presence of noise, the connection status of a link should rely on a hysteretic cycle: two nodes p and q are considered connected whenever their link have reached a value $l(p,q) \geq T_{CONN}$ and has not yet decreased below a hysteresis threshold T_{HYST}, with $T_{CONN} - T_{HYST}$ great enough to account for fluctuations of the l value induced by noise or transitory phenomena.

3.1.2 *The Distributed Algorithm.* For the actual execution of the distributed region formation algorithm, each node stores a vector describing, for each of its neighbors, the current value of the link l towards it and a flag signaling the status of that link (connected or not). The initial values of the l values can be arbitrary.

The distributed execution of the algorithm is based on a periodic scheme, which acts as a sort of continuous background activity in the sensor network, and does not require any

form of synchronization among sensors.

The basic scheme is as follows. Periodically, each node actively broadcasts to all its neighbors (i.e., all the sensors within its wireless radio range) a $DataExchangeRequest$ to request them a data exchange session. Then, with all the currently active neighbors (i.e., those who have answered to the ping request), it exchanges the data needed for computing the l values (e.g., the value of the property v upon which region formation relies) and executes the already described *UpdateLink* procedure for each of them.

Schematically:

```
Active Thread
    Do forerever {
        Wait(t);
        ActiveNeighbors[] = DataExchangeRequest();
        Foreach(ActiveNeighbors[]) {
            UpdateStatus {
                data = ExchangeData();
                UpdateLink(data);
            }
        }
    }
```

Clearly, such scheme also requires the instantiation in each node of a reactive thread in charge of reacting to incoming data exchange requests, exchange data accordingly and execute in its turn the $UpdateLink$ procedure. Schematically:

```
Passive Thread
    Do forerever {
        ListenDataExchangeRequests();
        UpdateStatus {
            data = ExchangeData();
            UpdateLink(data);
        }
    }
```

It is important to note that the data exchange and link updates (which we grouped under a single $UpdateStatus$ procedure) should be executed atomically at both sides, to ensure coherency of the perspective from both sides (i.e., two nodes have the same perception of the connection status of their link).

From the above description it is clear that our algorithm tends to impose a pre-defined, tunable, and uniform load, to the system (each node executes the same amount of operations). Consequently, it induces predictable and uniform energy costs across the whole system. Also, and very important, the proposed scheme does not necessarily require nodes to be active all the time.

To save energy, the scheme tolerates that nodes can be switched off periodically: over each period with length t, a node can be active (able to communicate with other nodes and, in particular, of reacting to data exchange requests) only a fraction $dutycycle$ of such time, while being switched off for the remaining *1 - dutycycle* fraction of the period. Since the nodes are assumed not to be synchronized with each other, the value of $dutycycle$ corresponds probabilistically to the fraction of active neighbors that will respond to a $DataExchangeRequest$ (see Figure 3). Of course, to avoid the situation in which a couple of neighbor nodes never happen to be active at the same time, some randomization of the t value around a mean t_{base} value is necessary.

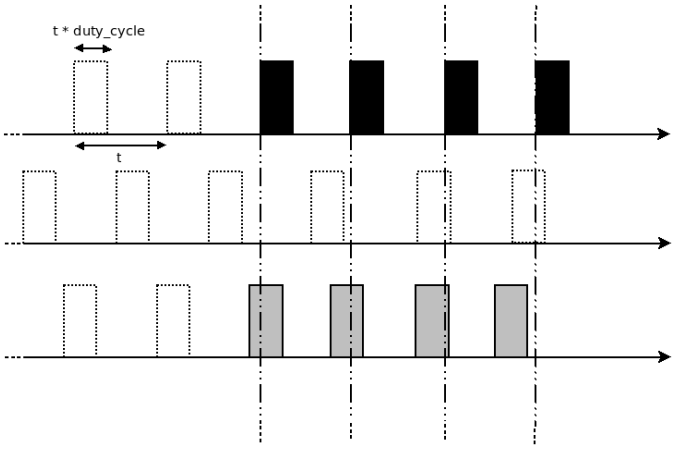


Fig. 3. The $dutycycle$ of nodes. The interval t determines the frequency of the node operations. The parameter $dutycycle \in [0, 1]$ establishes the fraction of t during which the node is active. Neighbor nodes can communicate if their active time is overlapping. For example the black node initially communicates with only the grey node while, after three iterations, it can only reach the white one.

In general, shorter t or higher $dutycycle$ tend to speed up the convergence of the algorithm, but increase the energy consumed by sensor per time unit. Therefore, every user or application can select the "degree of activity" of our algorithm and, so, the energy consumed over time. As a general rule, we can anticipate that: for macro virtual sensors to work well, t should be at least of the same order of magnitude of the sampling period of the phenomena to be measured by them; small values of $dutycycle$ (enabling a higher energy saving) are tolerable only in the presence of reasonably dense networks (and thus with high clustering [Albert and Barabasi 2002]).

Beside these factors, the actual spatial extension of the sensor network and its specific density are not very relevant in ViMS: since all the interaction are strictly local it can be applied also to (virtually) infinite and infinitely dense networks. These considerations are better substantiated in Section 4.

As a final remark, we are aware that enforcing some forms of global synchronization in the $dutycycle$ of the sensors (e.g., firefly inspired synchronization [Werner-Allen et al. 2005]) could have notably simplified our scheme, making it more energy efficient at nearly no additional costs. However, such an approach is generally not applicable when the sensors are to be directly queried by users. In fact, a global synchronization of the sleep-wakeup cycles of sensor would cause all sensors to be all sleeping at the same time, and thus unable to answer to users. Conversely, in our scheme, a user can always find, w.h.p., a close-by sensor to answer to his/her queries.

3.1.3 *Parameters Setting.* Let us now go into more details about the other parameters of our algorithm.

Concerning the parameter Δ, it determines how fast the link weight l changes its value and, consequently, how the region formation algorithm reacts to environmental changes. The choice of this parameter is not crucial, provided that it is chosen small enough to require several cycles of the $UpdateLink$ procedure to actually modify the status of links (and, thus, the shape of regions). In this way, the region formation algorithm avoids that

random or temporary fluctuations of the measured value at a node continuously induce changes in the established regions. For similar reasons, the $T_{CONN} - T_{HYST}$ hysteretic interval should be great enough with respect to Δ to avoid frequent changes to the connection status of two nodes.

Concerning T, i.e., the threshold that determines whether a link between two nodes should be strengthen or weakened, it is clear that this is a a domain dependent parameter. For example, consider a wildlife landscape where temperature is assumed as the value upon which to rely for region partitioning. There, a difference of 3°C can be considered relevant for a biologist to distinguish different ecological niches (e.g., a forest against a savanna), and (s)he could rely on a region-partitioning based on such a threshold. However, rangers may be interested in much higher thresholds (e.g., 30°C) to associate regions with potential anomalies such as fires. How the choice of the threshold can affect the region formation algorithm is exemplified in Figure 4: the finer the threshold, the more sensitive the region formation algorithm to the sensed environmental patterns, and the more and smaller regions tends to be formed.

In the vast majority of the cases, a domain expert can provide suitable and relevant thresholds to highlight the phenomena of interest and to drive the self-partitioning accordingly. In any case, it is worth emphasizing that our approach can easily enable the possibility to neglect the problem of some a priori threshold selection. In fact, the region formation algorithm can be exploited, at no additional communication/energy costs, to build any number of different overlay partitions, each based on different thresholds. Simply, each node can host and compute an array of virtual link values l for each neighbor, each corresponding to a different threshold value.

3.2 Aggregation

The ViMS approach assumes that each and every sensor within a region is locally provided with information related to the overall status of the region. To this end, it is possible to integrate forms of diffusive gossip-based aggregation [Corradi et al. 1999; Jelasity et al. 2005] within the described general scheme.

In simple terms, diffusive gossip-based aggregation works by having nodes periodically exchange with their neighbors the information about some local value, locally aggregate the values according to some aggregation function (e.g., maximum, minimum, average, etc.), and further exchange in the subsequent step the aggregated value. Eventually, the propagated aggregated values will diffuse over the whole network and will account for the aggregation function having been computed at each and every node of the network. Thus, each node of the network will have the local availability of the globally aggregated value.

3.2.1 *Global vs. Regional Aggregation.* In our approach, integrating general forms of diffusive gossip-based aggregation is extremely simple and incurs in nearly no additional communication costs. By considering the $UpdateStatus$ procedure that is executed at each data exchange session with one must:

—piggyback the data exchange message. Other than the data related to the v values upon which region formation rely, also the values in which one is interested to be aggregated;

—execute the aggregation function;

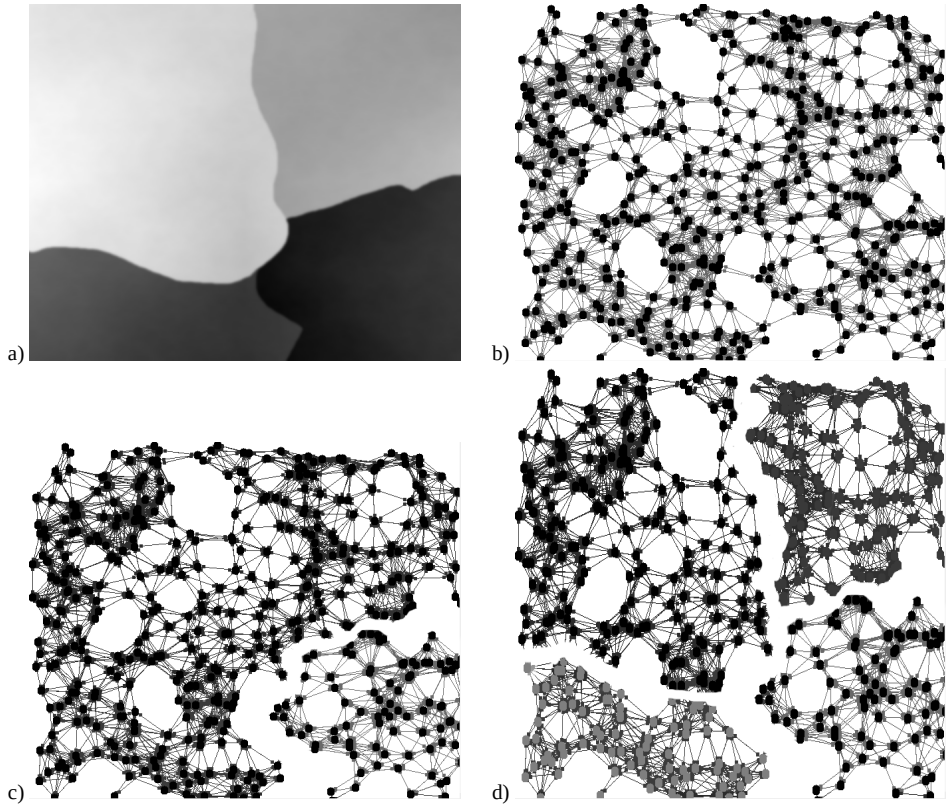


Fig. 4. Self-organizing spatial regions and the impact of the T threshold: a) a scalar field with 4 regions with different values of a property v varying from 0 (white) to 1 (black); b) a 500 nodes sensor network immersed in the above scalar field, with links representing the actual wireless connections among sensors; c) resulting overlay region organization when $T = 0.3$, leading to a partitioning into 2 coarse regions (we show only the logical links between the nodes that are logically connected); d) resulting overlay region organization when $T = 0.1$, leading to a partitioning into 4 small regions.

Schematically:

```
UpdateStatus
    data[] = ExchangeData();
    UpdateLink(data[RegionData]);
    Aggregation(data[AggrData]);
```

The above basic aggregation scheme leads to globally aggregate information over the whole sensor network, and does not account for the existence of regions. Of course, having the local availability of some global information may be of some use in various situations. However, globally aggregated values give very little details on the status of the network, are prone to obsolescence and high losses and are definitely of little use for users wishing to acquire information about environmental properties around him/her.

For these reasons, and because it is at the core of our approach, we mostly focus on per-region aggregation algorithms. Local, per-region, aggregation algorithms enable each sensor in a region to act as a sort of access point for aggregated data in that region, and thus realize the concept of virtual macro sensor: from the application viewpoint, one can perceive a region as including a single sensor covering the whole region.

When regions are already formed (transitory situations will be discussed later on), computing aggregation functions in a region reduces to executing the diffusive aggregation algorithm only between those couples of neighbor nodes that are in the same region (i.e., which are *connected*). Again, computing per-region aggregation functions does not introduce significant additional burden to the network. The exchange of the data to be locally aggregated can again occur by piggybacking over the existing messages, and the computation of local aggregation algorithms reduces to execute the aggregation function for those nodes that are connected with each other. Schematically:

```
UpdateStatus
    data[] = ExchangeData();
    UpdateLink(data[RegionData]);
    Aggregation(data[GlobalAggrData]); // if needed
    if(connected) {
        Aggregation(data[LocalAggrData]);
    }
```

Where the global aggregation function does not have necessarily to be included. We emphasize that the aggregation function (whether local or global) can apply to any of the values locally available at a node, other than to the value v upon which region formation relies. Let us now exemplify and discuss some relevant aggregation functions that have been integrated so far in ViMS.

3.2.2 *Minimum and Maximum.* The knowledge of the maximum and the minimum values of some properties over a region of a sensor network can be useful in many applications. As examples: a sensor can assess whether a reading is an outlier by comparing it with the maximum and the minimum, an expert can assess the presence of anomalies within the region.

To locally acquire the $max(w)$ and $min(w)$ values of some sensed value w, each node can simply exchange with its neighbors the data about the maximum and the minimum ($max(w)_p$ and $min(w)_p$, respectively) he locally knows so far. That is, a node p, after having exchanged such data with node q, executes the following *Aggregation* procedure:

Aggregation
if $(min(w)_p > min(w)_q)$ {
    $min(w)_p = min(w)_q$;
}
if $(max(w)_p < max(w)_q)$ {
    $max(w)_p = max(w)_q$;
}

with $min(w)_p$ and $max(w)_p$ both initialized at w_p. Eventually, the knowledge about the actual $min(w)$ and $max(w)$ will reach each node of the region (or, in the case of global aggregation, the whole network).

3.2.3 *Average.* Another very relevant aggregation function is the average, in that it can provide very compact information about the overall status of a region.

To compute the average, the aggregation function can simply average the local averages aggregated so far, as described in [Jelasity et al. 2005]. That is, a node p, after having exchanged data with connected node q, can simply execute the following aggregation function :

Aggregation
$avg(w)_p = (avg(w)_p + avg(w)_q)/2;$

with $avg(w)_p$ simply initialized at the local value w_p. Eventually, each node will have its local $avg(w)_p$ value converge to the actual average.

In ViMS, though, we have decided to adopt a modified algorithm for aggregating the average, to enable to properly deal with dynamics in regions and in the values being averaged (as discussed in Subsection 3.3). The rational behind such algorithm is to have each node reason on its local estimate of the average of a property w in terms of the differential with the actual local w value, i.e. for a node p.

$avg(w)_p = w_p + diff(w)_p;$

With $diff(w)_p$ being initialized at 0. Data exchange between two nodes p and q should include the exchange of both their w and $diff$ values. The aggregation function between these two neighbors should then mimic a sort of virtual load balancing action [Corradi et al. 1999], i.e., it must ensure that, for the values of $diff$ and w before (time t) and after (time $t+1$) the application of the aggregation function, the following are respected:

$$\begin{cases} w_p(t) + diff(w)_p(t+1) = w_q(t) + diff(w)_q(t+1) \\ diff(w)_p(t) + diff(w)_q(t) = diff(w)_p(t+1) + diff(w)_q(t+1) \end{cases}$$

where: the former expression states that the estimated averages at the two nodes have been equalized; the latter expression states that the differentials have been overall preserved. Resolving the expressions in $diff(w)_p(t+1)$, we obtain the following aggregation function:

Aggregation
$diff(w)_p = (diff(w)_p + diff(w)_q + w_q - w_p)/2;$
$avg(w)_p = w_p + diff(w)_p;$

It can be easily shown that by asynchronously applying the above calculation for subsequent couples of nodes in the system, whatever the order is, makes the average estimator $avg(w)_p$ at each node converge toward the global average of the value w in the region.

Also, it can be shown that the convergence of this aggregation function is not undermined by dynamic changes in the values w of the nodes.

3.2.4 *Other Aggregation Functions.* There are two additional aggregation functions worth to be discussed, due to their relevant in the ViMS approach, namely: region ID election and border-distance evaluation.

The former exploits a sort of per-region minimum identification towards the election of a region leader. By having each sensors exchange its unique ID with its neighbor, the minimum ID eventually recognized by each node will define the leader (and the leader itself will recognize itself as that). The identification of a regional leader is very important to give each macro sensor a recognizable unique identity.

The latter considers that each node at the border of a region (i.e., each node which has at least one virtual link l below the threshold) propagates within the region an hop counter initialized at 0. By having such counter re-propagated by each node on per-minimum basis, each node in a region eventually becomes aware of its distance form the closest border. This measure is important to asses, within regions, the sensing coverage of the macro sensors.

Also in these cases, computing such aggregation functions requires minimal additional effort by nodes, i.e., that of piggybacking a few more bits in the $ExchangeData$ messages.

When needed, one can also decide to exploit the same schema to compute any additional distributed aggregation algorithms (indeed, any aggregation function that is insensitive to order and duplication can be aggregated with the same basic scheme) as well as more detailed information about the structure of a region (e.g., the number of nodes in the region or some topological information about it [Zhu et al. 2008]).

3.3 Transitory and Dynamic Situations

In this section we analyze the behavior of ViMS in the presence of environmental dynamics and during transient states.

3.3.1 *Dynamics of Region Formation.* In general, the initial values of the virtual links l between nodes are irrelevant for region formation during the transitory phase after deployment. For instance, in an initial situation in which all nodes are disconnected from each other (l values all below the T_{CONN} threshold), each node represents a region in itself. As the algorithm starts running, nodes with similar values of v will start connecting with each other, and sets of regions with growing sizes will start forming and possibly merge each other until a stable situation. Vice versa, if all the nodes are initially connected, all the system represents a single huge region.

When the values v change, this can lead to a re-shape of the existing regions. For instance, a node p originally belonging to a region R_x can see the links connecting it to R_x weakens until it is no longer connected to a node in this region. Concurrently, it is possible that this node sees its links with the nodes in another region R_y to strengthen, until it eventually becomes part of R_y itself. During that process, it is possible that p is temporarily be detached from any region or that it is concurrently connected to R_x and R_y, thus making R_x and R_y a single region indeed. However, such situations does not cause any harm to the system, since they are temporary (and if they are not, it is simply because they should not be, e.g., the node p has to be a region in itself or region R_x and R_y must indeed merge into a single region).

3.3.2 *Regional Aggregation.* As far as regional data aggregation is concerned, the handling of transitory situations and of environmental dynamics is a bit more intricate.

At start up, when the system starts working right after deployment, the regional aggregation process can start concurrently with the region formation process. Let us firstly assume that nodes are initially not connected. Then, the local aggregation procedure starts actually taking place as soon as two nodes get connected in the same region, and it proceeds gradually involving more and more sensors, eventually converging towards a stable region situation. It can be shown (and it is quite intuitive indeed, due to the cumulative nature of aggregation) that the proposed aggregation algorithms do not experience problems if executed on a growing number of nodes, as it happens when regions are forming from a set of independent nodes. Simply, as new values to be aggregated enter into play, the very nature of diffusive gossip-based aggregation ensure that the new values will be eventually accounted in the overall aggregation process. Similar considerations apply to the case in which new sensors are dynamically deployed in a region of the system or whenever an existing region enlarges or merges with another region. The situation is a bit more complex in the presence of:

—shrinking regions, as it happens during a process of regions splitting, or simply when some sensors die;

—dynamic changes in the values to be aggregated, which is a normal situation indeed for a sensor network deployed in some environment to be monitored;

In both the above situations, the values computed so far by the local aggregation functions may no longer be valid, because they could have accounted for values that no longer pertain to the region. In the case of shrinking regions, for example, the former maximum may have left the region and/or the average may have notably changed. Similarly, in the case of dynamic changes in the values to be aggregated, the former maximum may have decreased and/or the average may have changed. Whatever the case, the inherently cumulative nature of aggregation does not enable the aggregated values to correctly reflect the new situation, which could induce notable errors in the resulting aggregated values.

A general and widely adopted solution to tackle this problem in aggregation relies on periodic restarts of the aggregation process [Jelasity et al. 2005]. Periodically, the nodes of a network (or of a region, in our case) start a new *epoch* of aggregation, by resetting previously aggregated values and re-starting the aggregation process. In this way, the errors cumulated so far are simply reset and the newly started aggregation process can account for the updated situation. Such epoch-based approach could be easily integrated in ViMS to deal with dynamics. Beside the fact that it requires a careful tuning of the epoch period, its key problem is that its periodic nature tends to produce notable inaccuracies in the estimated aggregated values.

3.3.3 *Evaporative Regional Aggregation.* We have identified a more general and more effective solution than the epoch-based one to deal with the dynamics of regions and of aggregated values.

Since the basic problem in properly handling regions and values dynamism is caused by the fact that values are monotonically cumulated during aggregation, we decided to compensate this by forcing a sort of *evaporation* of the values computed by the aggregation algorithms. In other words, the aggregated values at a node are slowly (compared to the convergence time of the aggregation algorithms) moved towards their initial values,

e.g., the local values of the node. In this way, the weight of those data cumulated by the algorithm will gradually diminish, unless properly re-enforced.

As an example, consider the case of the determining the maximum in a region. Let us assume that each node in a region has already locally available the value of such maximum, due to the aggregation process executed so far. Now, let us have each node slightly evaporate such value by making it diminish to approach the locally sensed value of the property. If the node holding such maximum is still in the region, the continuous execution of the aggregation process will make a node eventually receive again the maximum, thus undoing the evaporation effects. Instead, if the node holding the maximum left the region (or if its local value is no longer the maximum one in the region), evaporation will enable to stabilize the new maximum at each node.

Similar considerations apply, e.g., to the calculus of the average. However, to enable the evaporation process for the average, it is necessary that each node knows what to make evaporate to have the evaporation process slowly approach (if not re-enforced) the local value of the nodes. This is the reason why, in our average aggregation function (see Subsection 3.2) we had to evaluate the average by measuring it in terms of the difference with the local value of the node.

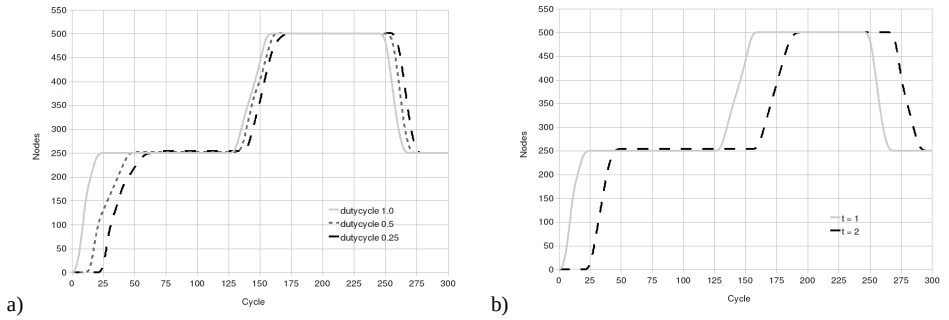
With respect to the general scheme of our algorithm, enforcing evaporation implies executing, at each execution of the $UpdateStatus$ procedure, an $Evaporate$ function that enforce evaporation of the values aggregated so far. Schematically:

UpdateStatus
data[] = ExchangeData();
UpdateLink(data[RegionData]);
Aggregation(data[GlobalAggrData]); // if needed
if(connected)
    Aggregation(data[LocalAggrData]);
Evaporate(GlobalAggrData); // if needed
Evaporate(LocalAggrData);

The $Evaporate$ function assumes the following forms for, e.g., the $max(w)$, $min(w)$, and $avg(w)$ aggregation functions of some property w:

Evaporate
$max(w)_p = MAX(max(w)_p - \Delta, w)$;
$min(w)_p = MIN(min(w)_p + \Delta, w)$;

if $(diff(w)_p < -\Delta)$ {
    $diff(w)_p += \Delta$;
} else if $(diff(w)_p > \Delta)$ {
    $diff(w)_p -= \Delta$;
} else {
    $diff(w)_p = 0$;
}



a) b)

Fig. 5. Region formation algorithm behavior. The increasing number of nodes of a growing region varying (a) $dutycycle$, $(usingt = 1)$ and (b) t, $(usingdutycyle = 1.0)$.

where Δ represents the value to be evaporated, which can be calculated as a small percentage of the difference between the actual value of a node and its current estimate of the aggregated value. As it will be shown in Section 4, any finer tuning of Δ (the only parameter of the evaporative approach, indeed) would not significantly affect the overall effectiveness of the aggregation in handling dynamics.

A similar evaporation-based approach can apply also to other aggregation function. However, as far as the re-determination of a new region leader once the former leader has left a region, a peculiar solution must be adopted (since the ID value cannot tolerate any actual evaporation in its value). Thus, each node will gradually evaporate its trust on the correctness of the value of the leader ID. Whenever such trust becomes excessively low, the current leader ID is considered obsolete and a new leader (i.e., the new node with the minimal ID) is identified and elected. However, since the leader ID has the only goal of enabling a unique identification of each macro sensor, but do not affect the way users can access data and their correctness, the process of "evaporating" and re-electing region leaders does not require any specific critical tuning of its parameters.

In summary, the proposed evaporative solution makes ViMS fully self-organizing and self-adaptable, and does not require careful tunings of the parameters involved in the process. Also, as quantified in Section 5, the proposed evaporative approach is generally more accurate than the epoch-based approach.

4. PERFORMANCE EVALUATION

To test the effectiveness of the approach, we have experimented it in both a simulation environment (to verify the convergence and accuracy level of our approach in large-scale scenarios, and the effect on them of the main parameters) and in a testbed (to evaluate its functioning in practice and its actual energy consumption).

This Section is organized as follows: in Subsection 4.1 we outline the qualitative performances of ViMS in terms of stability and convergence. In Subsection 4.2 we quantitatively evaluate the average ViMS accuracy. Finally, in Subsection 4.3 we discuss our experience with a real-world testbed.

4.1 Qualitative Behavior

4.1.1 *Region Formation.* ViMS has been simulated over the Repast framework. We have conducted several experiments with sensor networks of different sizes, immersed in

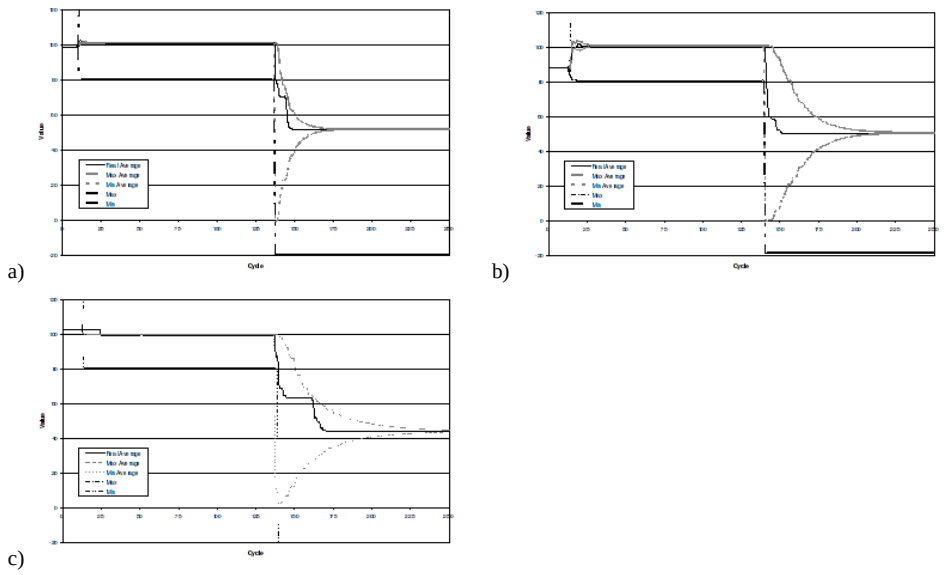


Fig. 6. Region aggregation algorithm behavior. Minimum estimate of the maximum, maximum estimate of the minimum, minimum and maximum estimates of the average and real value of the average with: (a) $dutycycle = 1.0, t = 1$, (b) $dutycycle = 0.5, t = 1$, (c) $dutycyle = 1.0, t = 2$.

different scalar fields, always obtaining similar qualitative and quantitative trends. The results reported here refer to: a scalar field with 4 recognizable spatial regions of similar sizes; a 500 nodes sensor network such that the average number of neighbors of each node is 15 (i.e., similar to the sensor network of Figure 4). Let us firstly analyze the behavior in region formation.

From a static viewpoint, simulations confirm that, as described in Subsection 3.1 and as shown in Figure 4, the region formation algorithm converges and variations on the parameter T actually induce the network in self-partition into regions of different sizes.

From the dynamic viewpoint, we have studied how $dutycycle$ affects the speed of convergence of the region detection algorithm. In particular we traced the evolution of the system with a changing environment. At startup, there is not any recognizable region. Nodes are thus not connected with any neighbor. Within cycles [0 – 128] the environment is partitioned into 2 large-regions of equal size. Between cycles [129 – 255] the environment is compacted into a single large region. Finally, at cycle 256 the environment is disaggregated again into two regions.

Figure 5-a shows the evolution in the average number of nodes in one region as time passes, by varying $dutycycle$. The graph shows that the number of nodes of the region start from 0, grow to 250 during the first phase [0 – 128 cycles], reaches 500 during the second phase [129 – 255 cycles], and then diminish again to 250. Not surprisingly, reducing $dutycycle$ makes the network proportionally slower in reaching convergence in region formation. Figure 5-b shows, as expected, that also t influences the converge speed of the region formation process.

It is fundamental to notice that when discussing gossip-based algorithms, like ViMS, the $dutycycle$ and the density of the network (average number of neighbors of a node)

are strongly related. To understand their relationship we must consider that the number of neighbors with which a node can interact is on average given by:

$interactingNeighbors = dutycycle^2 * density$

This is because $dutycycle$ can be considered the probability of a node being awake in a given interval (see Figure 3), and two nodes will be awake at the same time with $dutycycle$ probability. Thus the fraction of neighbor nodes awake at the same time is the one in the above formula. For example, with a $dutycycle = 1$, nodes stay awake all the time, thus each node will interact with all its neighbors (density nodes, by definition). With a $dutycycle = 0.5$, only a $1/4$ of all the neighbor nodes will be awake at the same time to interact, thus each node will interact on average with $density/4$ nodes.

Therefore, the behavior of a network with $dutycycle = 1$ and $density = 10$ would be equal to the behavior of a network with $dutycycle = 0.5$ and $density = 2.5(= 10/4)$. This implies that a varying density is equivalent to a varying $dutycycle$.

4.1.2 *Region Aggregation.* Here we consider the results of the aggregation produced by ViMS in all the regions in which the networks partitions.

From the static viewpoint we have of course verified that all local aggregation algorithms correctly converge towards the correct values.

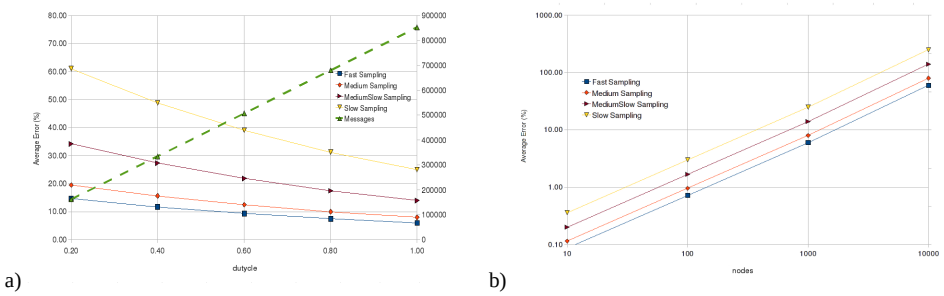
Figures 6-a-b-c show the trend of several values aggregated on a per region basis, in the first 250 cycles of the simulation scenario already discussed in the experiment setting of Subsection 4.1.1. Curves in each graph represent the minimum (worst case) estimate of the region maximum, the maximum (worst case) estimate of the region minimum, the minimum and the maximum (the two worst cases) estimates of the average, and the real actual value of the average computed over all nodes within the growing region. In all the graphs it is possible to see that ViMS eventually ensures that aggregated values converge to the new correct values upon changes.

All the graphs show the same qualitative trend. When regions start forming, after a few cycles, a fast convergence of the local maximum and minimum to their new correct values of 120 and 80, respectively, is clearly visible. Average related values have a relatively small transitory and eventually reach the correct value of 100 as expected. At cycle 128 the region starts growing another time. The local maximum does not have to change its value. The local minimum reaches quickly its new value (-20) in a few iterations. Average values instead have a longer transitory but eventually converge to the expected value of 50. Analyzing the convergence rate for different values of the sensor $dutycycle$. It is easy to see that the lower the $dutycycle$ (and thus the fewer the interacting nodes), the longer is the time to converge. This is rather intuitive: if only few nodes interact at a given cycle, the time to converge to a new value increases.

Figures 6-a-b-c illustrates also the effect of t on the convergence speed. Again, this is rather intuitive, the longer t, the longer the time to converge. This is because nodes sleep longer and thus some simulations cycles are “wasted” sleeping.

4.2 Accuracy

To understand if ViMS is able to produce reliable aggregated information especially in dynamic environments, we defined a metric to measure its accuracy. Moreover, through several experiments explained in the following, we studied how ViMS’s overall accuracy is influenced by $dutycycle$ and by the $size$ of the regions. As far as environmental dynamism is concerned, we tested the accuracy of ViMS in 4 different working conditions. To define



a) b)

Fig. 7. Aggregated ViMS performances. Average error and number of exchanged messages in different environmental settings, varying (a) $dutycycle$, (b) the size (number of nodes) of a region.

these 4 settings, we choose a phenomenon which need a known sampling rate (SR) to be properly monitored. Then we conducted experiments using:

$t = 1/3SR(fastsampling)$
$t = 1/SR(mediumsampling)$
$t = 2/SR(mediumslowsampling)$
$t = 3/SR(slowsampling)$

For each sampling speed we conducted several experiments varying the $dutycycle$ parameter to verify its impact (along with the impact of density) on accuracy, and also varying the size of the region in which accuracy is measured, to assess the scalability of ViMS. The accuracy of ViMS was measured as follows:

$$\sum_{time=1}^{T} \frac{max(avg(v)(t)) - ravg(v)(t)}{ravg(v)(t)}$$

This error represents the normalized difference between the maximum estimate of the average $max(avg(v)(t))$ and the real average $ravg(v)(t)$. This is a good synthetic index to understand how good ViMS is able to perform and gives a precise idea of the "worst case" error normalized over time.

Figure 7-a shows, as expected, that under all the sampling conditions, the average error increases by decreasing the $dutycycle$ and the sampling frequency. This behavior can be explained by the fact that, in both the cases, the number of gossip interaction decreases and, thus, also the converge speed of the algorithm. The less the convergence speed, the more the errors. Moreover it is worth noting that to reach high precisions it is important to choose a proper sampling rate. In fact the errors quickly step up while choosing a sampling rate lower that the proper one *(Medium Sampling)*. Finally, the number of messages being exchanged (that obviously do not depend on the environment) increases with the $dutycycle$.

Figure 7-b shows the scalability of our approach, under the conditions we tested it, in regions up to 10000 nodes. The results show a linear trend. Considering that sensor networks up to now are composed by no more than few hundreds of nodes, we consider them encouraging. Moreover, even considering a medium-term scenario pervaded by sensor networks with many thousands of nodes we think that the ViMS approach could be useful as well. In fact we think that despite huge sensor networks, the region partitioning system

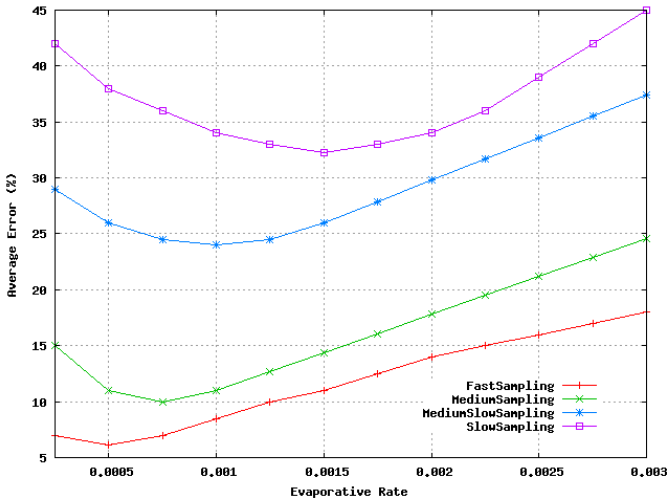


Fig. 8. The effects of the evaporative rate over the ViMS(EVP) approach.

embedded into ViMS will maintain the regions' size manageable.

ViMS performances are also somewhat influenced by the evaporative rate Δ. For each possible value of t we ran several simulation using different values of Δ. Figure 8 shows the relation between the average error rate and Δ. It is clear from the graph that for every t, an optimum value of Δ which produces the lowest error exists. Furthermore, it is clear that high values of Δ are required to compensate low sampling frequencies. This happens because a more dynamic environment requires an higher evaporative rate. The bigger fluctuations, the bigger Δ should be used to follow them. However, it is also interesting to note that the performances of ViMS are not that greatly dependent over the value of Δ chosen. In fact, even if ViMS is not able to self-adapt the values of Δ, changes in its value do not produce much greater errors over the optimal value. In any case, the possibility to integrate a mechanism to autonomically adapt the value of Δ to the dynamics of the environment could be worth investigating.

4.3 ViMS Test bed

To verify the feasibility and the actual energy consumption of ViMS, we have deployed a testbed of 16 Micaz Crossbow integrating the ViMS algorithms across two confining rooms and the facing corridor of our department (see Figure 9). Light levels have been used as the basis for region identification, whereas both light and temperature have been aggregated on a per-region basis. This testbed has been used to enable a mobile user equipped with a laptop and a Crossbow board to query multiple sensor and retrieve from them aggregated data about their region.

The deployed algorithms worked as expected from the functional viewpoint. First, the different light levels of the three rooms led the sensor network self-partition in three different regions (i.e., three macro sensors), each associated to a different room. Second, within each region, the sensors correctly computed aggregate light and temperature information. Third, users were able to access such aggregated information by querying any sensor in a region as if it were a virtual macro sensor representative of its region.

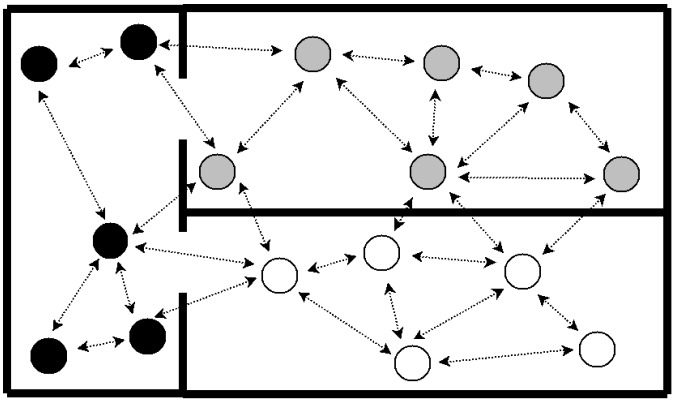


Fig. 9. Real testbed deployment. The 3 rooms identify 3 regions in the sensor network.

	t = 30s	t = 60s	t = 120s
dutycycle = 1.00	1.00x	1.31x	1.48x
dutycycle = 0.50	2.12x	2.68x	3.87x
dutycycle = 0.25	4.23x	5.97x	7.04x

Table I. Network life time under different settings. The worst case ($dutycycle = 1.0, t = 30sec$) represents approximately four days.

From the energy consumption viewpoint, setting $t = 30$ sec and $dutycycle = 1.0$, the algorithm drained a couple of AA batteries in approximately four days. On the opposite side, setting $t = 120$ sec and $dutycycle = 0.25$ the system lasted almost one month. In any case, we must consider that we MicaZ motes we have used are not highly optimized for low energy footprint, and much large durations can be expected with more recent sensor architectures. Table I summarizes the mean values of the energy used by sensors in terms of the ratio between the average sensor life and the shortest lasting case. Different results for different values of the parameters t and $dutycycle$ confirm that both these parameters can be effectively used to tune the energy consumption. In addition, the fact that the data are approximately constant over several experiments and with different nodes' configurations confirms that our approach is able to establish a well defined bound on energy consumption. Moreover it is important to remember that these results are independent from the monitored environment's dynamics. It is also worth noting that the sensor's life is linearly dependent from both $dutycycle$ and t. Particularly, it is related with $dutycycle$ by a factor 2, while with t by a factor 1.3. This fact is related to the hardware architecture of the sensors available today. The energy cost to keep the radio listening is very high compared to the cost of internal computation. An interesting property of our system is that the sensors' energy consumption is uniform over the network.

It is finally important to compare Table I with the simulated results in the previous Subsections. With our approach, it is possible to tune the desired energy consumption by acting both on t and $dutycycle$. The cost to be paid for a lower energy consumption consists in a slower converge speed and reduced accuracy in aggregation.

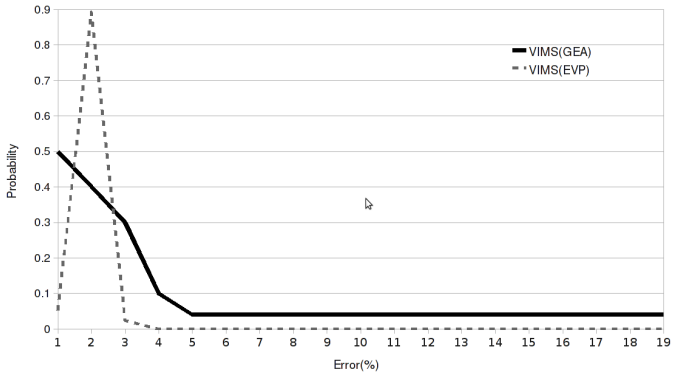


Fig. 10. Comparison of the density of errors in aggregated average computation.

5. COMPARISON WITH OTHER APPROACHES

To further verify the effectiveness of ViMS, we also compared it with two alternative approaches. Firstly, we have compared the effectiveness in handling dynamics of the evaporative approach integrated in ViMS against the epoch-based approach based on periodic restarts. Secondly, we have compared the ViMS approach with a reactive tree-based approach. These comparative analysis also enable to highlight and discuss the conditions in which ViMS can be effectively applied.

5.1 Evaporation VS. Epoch-based Aggregation

Epoch-based aggregation is an alternative approach to handle dynamics in diffusive aggregation. Instead of slowly evaporating aggregated data, one can periodically restart the aggregation process.

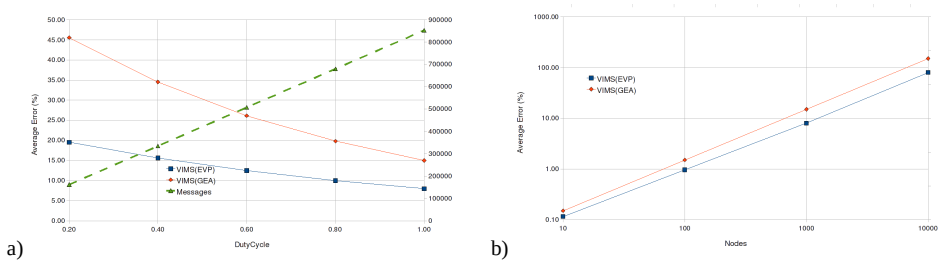
To compare evaporation with epoch-based aggregation, we simulated a system able to aggregate both global and regional data starting from the epoch-based algorithm proposed in [Jelasity et al. 2005]. In the following of the paper we will refer to it as VIMS(GEA) (Gossip-based Epoch Aggregator) while we will use ViMS(EVP) for the evaporative version.

The main difference between ViMS(EVP) and ViMS(GEA) is that ViMS(GEA) exhibits a periodic trend of fluctuations in the error. This is produced by two principal phenomenons:

—ViMS(GEA) can produce huge errors until it does not converge properly. While at the very end of an epoch ViMS(GEA) usually produces slightly lower errors, ViMS(EVP) performs much better when the epoch is still far from converging.

—ViMS(GEA) may produce huge errors even at the end of an epoch. If fact, if for example the new epoch is not started immediately after the previous one, environmental changes are not considered and aggregated values age quickly producing errors.

The error produced by ViMS(EVP) is much more stable and predictable, in that it is based on continuous operations rather than periodic restarts. To confirm this idea we plotted in Figure 10 the probability density of the average error along thousands of simulation ticks. The graph shows that almost all the errors produced by ViMS(EVP) are rather small (close to 2%). The errors in ViMS(GEA) are instead more sparse (most of errors vary in



a) b)

Fig. 11. Comparison graph between ViMS(EVP) and ViMS(GEA) for different (a) values of dutycycle and (b) number of nodes.

the range of 0%-5% of the real value). In general, it is clear that evaporation exhibits a lower variance and thus is more reliable.

In Figure 11 we compare ViMS(EVP) and ViMS(GEA). It reports the average error and the number of messages being exchanged by the two algorithms to compute the average value of the scalar field sensed by a sensor network. Figure 11-a considers different values of $dutycycle$ while Figure 11-b different network sizes.

In both the cases it is easy to see that the number of messages being exchanged is the same. This is rather obvious both the two aggregation approaches relies on ViMS and aggregation data are only piggy-backed on the same gossip protocol. More interestingly, it is possible to see that ViMS(EVP) performs 5-10%, in terms of average accuracy, better than ViMS(GEA). This is because the periodic restarts of ViMS(GEA) rise the error during the convergence and divergence phase.

5.2 ViMS VS. Routing Tree Algorithms

It is fundamental to compare ViMS with an approach providing direct access to sensor data via the construction of spanning routing trees as this and has been the prominent approach in the area. One of the most recent and efficient tools in this area is Scenes [Kabaday and Julien 2007].

Pervasive, immersed applications demand opportunistic and unpredictable interactions with local devices. This pervasive computing driven perspective demands communication abstractions that enable the required direct communication among mobile applications and embedded sensors. The Scene abstraction allows immersed applications to create dynamic distributed data structures over the sensor network. A Scene is created based on application requirements, properties of the underlying network, and properties of the physical environment.

Basically, once a user looks for data concerning a given area, Scenes dynamically creates a routing tree with the root (sink) where the user is located and spanning the sensors in the area. Sensor readings are then routed to the user via the tree. While the standard tree-based approach has a fixed sink where the data of all the network is collected, Scenes dynamically creates such trees in response to user queries. Accordingly, Scenes best suits the pervasive and mobile scenario we are dealing with and thus we have chosen it as a benchmark.

To conduct our comparison, we did not simulated the system in every detail, but only those parts needed to estimate the overall number of exchanged messages to construct the tree and to collect data from it.

In particular we conducted some experiments to test this approach and verify the number

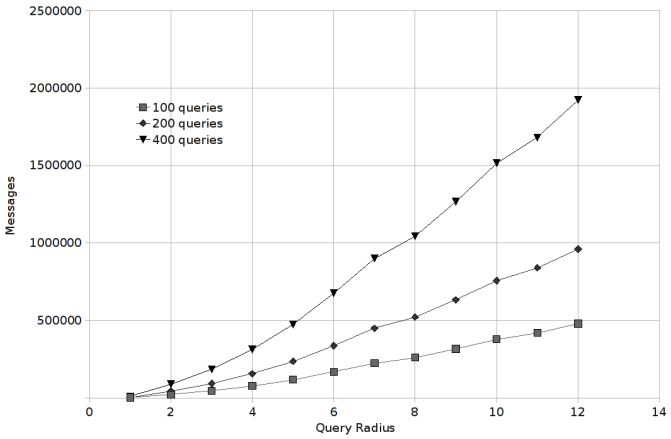


Fig. 12. Aggregated tree-based algorithm performances. Power consumption is linearly dependent on the number of queries, while it is quadratically dependent on the radius. On the opposite side ViMS power consumption depends only on the chosen accuracy level and would be represented by straight horizontal lines.

of messages required by the algorithm to build trees. For the sake of visualization we report results in the case of 100, 200, 400 queries having a radius from 1 to 12 hops form the source. Presented results are averaged over 200 experiments (see Figure 12). The number of messages being exchanged is independent on the time frame during which the queries are performed. It is possible to see that the number of messages and thus the power consumption is linearly dependent on the number of queries while it is quadratically dependent on the radius. This is because the area covered by the query and thus the number of sensors increases with the square of the radius, while queries are independent from each other and thus add up linearly. Moreover it is worth noting that this behavior is the opposite of ViMS. In fact, ViMS power consumption is completely unrelated with the number and the size of the queries and only depends on the chosen accuracy level.

5.3 Summary

To summarize this work, we built a comparison chart that shows which approach among evaporative ViMS *VIMS(EVP)*, epoch-based ViMS *VIMS(GEA)* and *Routing Trees*, best suits in different situations. For the sake of clarity, we selected 9 representative scenarios on the basis of the number of queries the user wants to perform for each time unit and of the size of the region spanned by the query. In particular, we considered scenarios in which the user wants to perform 1, 2 or 3 queries (with an average radius of 5, 10 or 15 hops) every period of $3t$.

For each scenario we drew a matrix showing for slow-, medium-, and fast-sampling which is the cheapest approach that can guarantee an average error rate under a certain tolerable threshold. (see Figure 13).

To understand how this matrix works we start from a very general discussion. There are three main trends in the matrix:

—For small query radius (5 hops) routing trees are almost always the best solution. This is because the energy cost of constructing and routing data in a small tree on response to a user query is much lower than the constant energy cost of ViMS. On the contrary,

for long-range queries (15 hops) ViMS (in both EVP and GEA flavors) is almost always the preferred solution. The same trend can be seen with regard to the number of queries. With a large number of queries ViMS becomes more cost effective.

—Focusing on a single chart of the matrix (e.g., 1st row, 2nd column) it is clear that the more error the application can tolerate, the more ViMS becomes the way to go. This is rather straightforward: we can slow down gossip interactions, saving energy, to match the requested error level.

—Focusing again on the whole picture in is clear that, in general, when slow-sampling speeds are concerned ViMS has an advantage over Routing Trees. This is because, even if its accuracy tends to decrease, its power consumptions is able to reach very low levels.

Let us introduce a more detailed analysis with a quantitative scenario. Consider a crowded square in a modern city in the near future. A sensor network is deployed on it with an average diameter of 15 hops. Without any loss of generality we can consider the whole network as un-partitioned (i.e., as a single region). Some people, let's say 150 each day, will query the digital infrastructure in place for getting the fastest direction where to go. The people density can be treated as a three dimensional function expressing the average density on a square meter. This function has a lower bound of 0 and several local minimum and maximum values. Usually, the global average of this function will change from a maximum of 4 persons per m^2 around midday to a minimum close to 0 during the night.

Let us imagine that the city's administration – controlling the sensor network – is, for example, interested in a very long lasting installation. Sensors are thus programmed to measure the environment every 10 minutes. Let us assume now that a 30% error can be tolerated. Under these conditions the sampling frequency can be considered slow. Given the above conditions, circa 1 query per t and 15 hops it is possible to see that we are dealing with the element (3,3) in Figure 13.

Since, we are dealing with a slow environment and we can tolerate an error of 30%, then the best approach is ViMS(EVP). On the contrary, if we were interested in only 5 hops, routing trees would be the best option.

From our perspective, comparison results, like those in Figure 13, are very important. In that they provide a sort of blueprint to deploy a sensor network in real scenario. On the basis of required monitoring conditions, the chart give concrete guidelines on the kind of algorithm to use.

6. APPLICATION SCENARIOS AND LIMITATIONS

The ViMS approach can be fruitfully exploited in a number of applications scenarios, although it still exhibits a number of limitations calling for further research work and extensions.

6.1 Querying by Multiple and Mobile Users

The ViMS infrastructure and its API best support the localized queries by multiple and mobile users. User that wants to retrieve information about the surrounding will typically access the nearest sensor and query it about some local patterns of sensed data. For example, "give me the maximum temperature within 500 meters" or, by referring to some more logical environmental concept, "give me the average temperature in this room". At this point, in most of the cases, the queried sensor can immediately answer to the user without

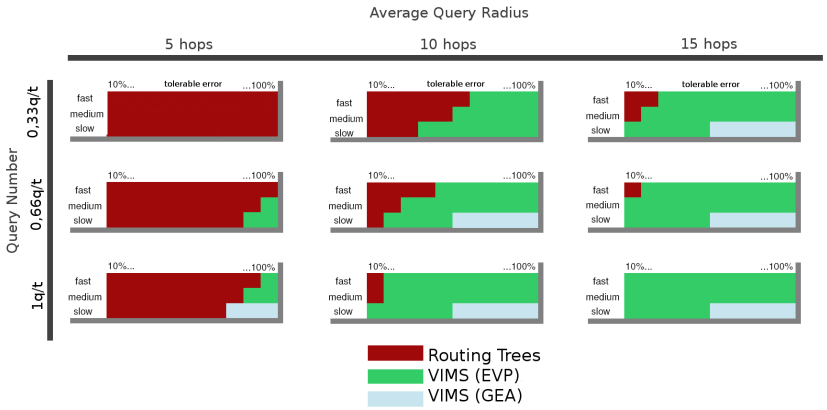


Fig. 13. Chart to compare Routing Trees, ViMS(EVP) and ViMS(GEA). Each of the 9 subgraphs represents the comparison for a given Average Query Radius and a given Query Number. Each subgraph consists of 3 lines, each one representing a different sampling speed. Each line shows the best (most energy efficient) approach for a given tolerable error.

further burdening the network, independently of the number of mobile users. In general, ViMS can answer to user queries whenever:

—the query relates to some functions of a property v which have been already aggregated as part of the background aggregation noise;

—the query concerns information within a single region, i.e., a single macro sensor;

With this regard, we recall (Subsection 3.2) that each sensor in a region knows its distance to the closest border of the region. Therefore it can recognize whether a query can be answered within the region because involving a single macro sensor (e.g., “give me the data in this room” or “give the data within a distance that is less then you distance to the border”) or not.

However, there are other kind of queries that are much more difficult to be answered in ViMS. One the one hand, it is necessary to allow computing additional aggregation functions other than the ones we have discussed. On the other hand, ViMS does not support querying distant sensors (i.e., sensors outside the user region) in that it does not provide a long-range routing mechanism. To solve these issues we might take the following actions. To make virtual macro sensors able to answer to very general queries related to any function of any property w sensed within a region, our approach should be extended to make macro sensors programmable. This implies the possibility of dynamically “injecting” into a sensor network the specification of additional local aggregation functions, and let these be integrated into the existing background aggregation noise. This can make it possible – within the sensing capabilities of individual sensors – to have macro sensors able to answer to both general-purpose and application-specific queries, as proposed by e.g. TeenyLIME

[Costa et al. 2007] and Logical Neighborhoods [Mottola and Picco 2006]. Clearly, such way of injecting new behaviors into the network could also be used to dynamically enforce a region partitioning based on more than one single property and on different distance functions (region partitioning based on different threshold values for a single property has already been discussed in Section 3).

To make virtual macro sensors able to answer also to inter-region queries (e.g., to let a macro sensor answer about "the average temperature within 500 meters" even if the query is performed at a distance of 300 meters from the confining regions) without falling back to a raw tree-based approach, we are currently verifying the possibility of implementing efficient inter-region aggregation algorithms based on gossiping. This will make any node in a region able to provide users with some aggregate information related to neighboring regions other than to its own region.

6.2 Centralized Data Collection

As already anticipated, the expected dramatic increase in the number and density of sensor networks deployed in our world, will soon reach a point in which the overall amount of data generated by such network will make it impossible to transfer these data to some centralized location in a raw way. Collection of aggregated data will become the only solution to extract useful information from them. The virtual macro sensors approach already solves this problem. In fact, it is possible to route aggregated macro sensorial data periodically to a fixed sink at very limited communication costs, simply by having the leader of each region take care of this alone. This can take place with the additional advantages that collected data fully abstracts from the structure and dynamics of the sensor network, and that local aggregation limits the loss of information which is instead associated with global aggregation algorithms that do not take into account data patterns.

However, to make the aggregated data collected at a centralized point really meaningful, it should be also possible to associate a specific dimension and shape to each region, and it should be possible to put regions in proper spatial relations with each other, so as to actually perceive the "network of macro sensors". More in general, the ViMS approach represents a mechanism to extract and collect semantic knowledge about an unknown environment [Bicocchi et al. 2009]. To make an example, one could think at randomly deploying sensors over some labyrinthine environment, have the sensor network self-organize into macro sensors based on light and/or sound patterns and then, if each region could recognize and report about neighboring regions, dynamically reconstruct a map of the area and of its salient environmental characteristics.

So far, we have experienced only the possibility to compute the distance of a sensor from the region border. Yet, we expect similar diffusion algorithms (possibly exploiting the already mentioned inter-region aggregation algorithms) can be defined to compute more advanced topological properties, and to extract high-level knowledge from a network. These, together with the mentioned possibility of enforcing multiple partitioning based on different properties and thresholds, could enable to produce, within the same network, multiple knowledge views for the use of diverse users/applications.

6.3 Situation Recognition

The possibility of identifying regions characterized by specific patterns of sensed data, and the possibility of computing aggregated data within the network can also be effectively used to improve the capability of the sensor network to self-recognize unusual patterns of

sensing and, in case, to automatically generate alarms.

To make a practical example, we are cooperating with the geological department of the Reggio Emilia Apennines to exploit ViMS for landslide detection [Rosi et al. 2007]. Inertial sensors deployed on a mountain slope generally sense a random background noise, without any recognizable patterns. However, when a slip surface starts detaching, all the sensors on such surface will exhibit peculiar acceleration patterns. In this case, our algorithm can be able – despite noisy data – to dynamically self-partition the network into two distinct regions, one of which associated the slip surface, and to alert the geological department by reporting macro sensorial information about the slip surface and its behavior. This can avoid the costly process of continuously reporting data back to the department for offline analysis. Other examples include detecting anomalies in buildings, streets, or parks. Also in this case, to make situation recognition fully practical and general purpose, both algorithms for dynamic injection of aggregation functions and algorithms for the recognition of advanced topological properties may be required.

Another important issue that should deserve further research is to develop a system to automatically monitor and analyze the usage patterns of the sensor network. On the basis of such analysis it would be possible to create a fully autonomic system able to switch between ViMS or routing-tree data collection on the basis of the actual demands to the network. For example, if the system detects the presence of a fixed node performing a lot of queries, it could automatically create a routing tree to serve that node, while adopting ViMS to serve other mobile nodes in the area.

7. CONCLUSIONS AND FUTURE WORK

The proposed virtual macro sensor approach makes a sensor network self-organize into regions characterized by similar sensing patterns, so as to promote aggregation of data on a per-region basis, as if each region were monitored by a single macro sensor. Such an approach can be very effective in supporting multiple and mobile users, in facilitating data collection in dense large-scale sensor networks, and in enforcing situation recognition activities.

Despite the encouraging results obtained so far, we are aware of some limitations of this work which are currently subject of research. Specifically we are aiming to generalize the approach to support multiple overlays and general-purpose queries and to explore inter-region algorithms to support queries involving multiple macro sensors. Finally, we will explore how the presented mechanism can apply to social sensor scenarios, in which users with mobile terminals *are* the sensors providing information about their environment via social networking tools (e.g., Twitter) [Sakaki et al. 2010].

ACKNOWLEDGMENTS

The material here presented is based on a research funded by the CASCADAS (*Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services*) project EU FP6-027807, which authors are glad to acknowledge.

REFERENCES

ABDELZAHER, T., ANOKWA, Y., BODA, P., BURKE, J., ESTRIN, D., GUIBAS, L., KANSAL, A., MADDEN, S., AND REICH, J. 2007. Mobiscopes for human spaces. *IEEE Pervasive Computing 6,* 2, 20 – 29.

ALBERT, R. AND BARABASI, A. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics 74,* 47, 47 – 97.

BANDINI, S., MAURI, G., PAVESI, G., AND SIMONE, C. 2001. Parallel simulation of reaction-diffusion phenomena in percolation processes: A model based on cellular automata. *Future Generation Computer Systems 17*, 6, 679 – 688.

BAUMGARTEN, M., BICOCCHI, N., MULVENNA, M., AND ZAMBONELLI, F. 2006. Self-organizing knowledge networks for smart world infrastructures. In *International Conference on Self-organization in Multiagent Systems*. Erfurt, Germany.

BICOCCHI, N., BAUMGARTEN, M., KUSBER, R., MAMEI, M., MULVENNA, M., AND ZAMBONELLI, F. 2009. Self-organizing data ecologies for pervasive autonomic services: the knowledge networks approach. *IEEE Transactions on Systems, Man, and Cybernetics*.

BOULIS, A., GANERIVAL, S., AND SRIVASTAVA, M. 2003. Aggregation in sensor network: An energy-accuracy trade-off. In *International Workshop on Sensor and Actuator Network Protocols and Applications*. Anchorage (AK), USA.

CASTELLI, G., ROSI, A., MAMEI, M., AND ZAMBONELLI, F. 2007. A simple model and infrastructure for context-aware browsing of the world. In *IEEE International Conference on Pervasive Computing and Communication*. New York (NY), USA.

CATTERALL, E., LAERHOVEN, K., AND STROHBACH, M. 2002. Self-organization in ad-hoc sensor networks: An empirical study. In *International Conference on Simulation and Synthesis of Living Systems*. Sydney, Australia.

CORMODE, G., GAROFALAKIS, M., MUTHUKRISHNAN, S., AND RASTOGI, R. 2005. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *ACM International Conference on Management of Data*. Baltimore (MD), USA.

CORRADI, A., LEONARDI, L., AND ZAMBONELLI, F. 1999. Diffusive load balancing policies for dynamic applications. *IEEE Concurrency 7*, 11, 22 – 31.

COSTA, P., MOTTOLA, L., MURPHY, A., AND PICCO, P. 2007. Programming wireless sensor networks with the teenylime middleware. In *ACM Middleware Conference*. Newport Beach (CA), USA.

CURINO, C., GIANI, M., GIORGETTA, M., GIUSTI, A., MURPHY, A., AND PICCO, G. 2005. Mobile data collection in sensor networks: The tinylime middleware. *Journal of Pervasive and Mobile Computing 4*, 1, 446 – 469.

DIMAKIS, A., SARWATE, A., AND WAINWRIGHT, M. 2006. Geographic gossip: Efficient aggregation for sensor networks. In *International Conference on Information Processing in Sensor Networks*. Nashville (TN), USA.

GEHRKE, J. AND MADDEN, S. 2004. Query processing in sensor networks. *IEEE Pervasive Computer 3*, 1, 46 – 65.

JELASITY, M. AND KERMARREC, A. 2006. Ordered slicing of very large-scale overlay networks. In *IEEE International Conference on Peer-to-Peer Computing*. Cambridge, UK.

JELASITY, M., MONTRESOR, A., AND BABAOGLU, O. 2005. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems 23*, 3, 219 – 252.

JIANG, X., POLASTRE, J., AND CULLER, D. 2005. Perpetual environmentally powered sensor networks. In *International Symposium on Information Processing in Sensor Networks*. Los Angeles (CA), USA.

KABADAY, S. AND JULIEN, C. 2007. Scenes: Abstracting interaction in immersive sensor networks. *Journal on Pervasive and Mobile Computing 3*, 6, 635 – 658.

KHELIL, A., SHAIKH, F. K., ALI, A., AND SURI, N. 2009. gmap: Efficient construction of global maps for mobility-assisted wireless sensor networks. In *International Conference on Wireless On-Demand Network Systems and Services*. Snowbird (UT), USA.

LOTFINEZHAD, M., LIANG, B., AND SOUSA, E. 2008. Adaptive cluster-based data collection in sensor networks with direct sink access. *IEEE Transactions on Mobile Computing 7*, 7, 884 – 897.

LU, C., XING, G., CHIPARA, O., FOK, C., AND BHATTACHARYA, S. 2005. A spatiotemporal query service for mobile users in sensor networks. In *International Conference on Distributed Computing Systems*. Columbus (OH), USA.

MADDEN, S. AND HELLERSTEIN, J. 2002. Distributing queries over low-power wireless sensor networks. In *ACM International Conference on Management of Data*. Madison (WI), USA.

MOTTOLA, G. AND PICCO, G. P. 2006. Logical neighborhoods: A programming abstraction for wireless sensor networks. In *IEEE International Conference on Distributed Computing in Sensor Systems*. San Francisco (CA), USA.

MULLER, R. AND ALONSO, G. 2005. Shared queries in sensor networks for multi-user support. In *Technical Report 508*. ETH, Zurich.

NEWTON, R. AND WELSH, M. 2004. Region streams: Functional macroprogramming for sensor networks. In *International Workshop on Data Management for Sensor Networks*. Toronto, Canada.

PANANGADAN, A. AND SUKHATME, G. 2005. Data segmentation for region detection in a sensor network. In *Technical Report 05-005*. University of Southern California (CA), USA.

POLASTRE, J., SZEWCYK, R., MAINWARING, A., CULLER, D., AND ANDERSON, J. 2004. Analysis of wireless sensor networks for habitat monitoring. 399 – 423.

RAMANATHAN, N., BALZANO, L., ESTRIN, D., HANSEN, M., HARMON, T., JAY, J., KAISER, W., AND SUKHATME, G. 2006. Designing wireless sensor networks as a shared resource for sustainable development. In *International Conference on Information and Communication Technologies and Development*. Berkeley (CA), USA.

RIVA, O. AND BORCEA, C. 2007. The urbanet revolution: Sensor power to the people! *IEEE Pervasive Computing 6*, 2, 44 – 44.

ROSI, A., MAMEI, M., AND ZAMBONELLI, F. 2007. Landslide monitoring with sensor networks: a case for autonomic communication services. In *Conference on Wireless Rural and Emergency Communications*. Rome, Italy.

SAKAKI, T., OKAZAKI, M., AND MATSUO, Y. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *International World Wide Web Conference*. Raleigh (NC),USA.

SARKAR, R., ZHU, X., AND GAO, J. 2007. Hierarchical spatial gossip for multi-resolution representations in sensor networks. In *International Conference on Information Processing in Sensor Networks*. Cambridge (MA), USA.

SCHOELLHAMMER, T., GREENSTEIN, B., AND ESTRIN, D. 2006. Hyper: A routing protocol to support mobile users of sensor networks. In *International Symposium on Mobile Ad Hoc Networking and Computing*. Florence, Italy.

SKRABA, P., FANG, Q., NGUYEN, A., AND GUIBAS, L. 2006. Sweeps over wireless sensor networks. In *International Conference on Information processing in Sensor Networks*. Nashville (TN), USA.

WANT, R. AND PERING, T. 2005. System challenges for ubiquitous and pervasive computing. In *ACM International Conference on Software Engineering*. St. Louis (MO), USA.

WERNER-ALLEN, G., TEWARI, G., PATEL, A., WELSH, M., AND NAGPAL, R. 2005. Firefly-inspired sensor network synchronicity with realistic radio effects. In *ACM International Conference on Embedded Networked Sensor Systems*. San Diego (CA), USA.

YANG, H., YE, F., AND SIKDAR, B. 2008. A swarm intelligence based protocol for data acquisition in networks with mobile sinks. *IEEE Transactions on Mobile Computing 7*, 8, 931 – 945.

YOUNIS, O., FAHMY, S., AND SANTI, P. 2005. An architecture for robust sensor network communications. *International Journal of Distributed Sensor Networks 1*, 3, 305 – 327.

ZHU, X., SARKAR, R., GAO, J., AND MITCHELL, J. 2008. Light-weight contour tracking in wireless sensor networks. In *IEEE Conference on Computer Communications*. Phoenix (AZ), USA.