Self-organizing Spatial Regions for Sensor Network Infrastructures

Nicola Bicocchi, Marco Mamei, Franco Zambonelli

DISMI – Università di Modena e Reggio Emilia – Via Amendola 2 – Reggio Emilia – ITALY
{nicola.bicocchi, marco.mamei, franco.zambonelli}@unimore.it

Abstract

This paper focuses on sensor networks as shared environmental infrastructures, and presents an approach to enable a sensor network to self-partition itself, at pre-defined energy costs, into spatial regions of nodes characterized by similar patterns of sensed data. Such regions can then be used to aggregate data on a per-region basis and to enable multiple mobile users to extract information at limited and pre-defined costs.

1. Introduction

The increasing diffusion of sensor networks [3] will make them become more and more as a general shared infrastructure for the use of multiple [4, 10] and possibly mobile users [4, 7]. In the near future, users in an environment will be able access the sensors in their neighborhood to gather information about the surrounding physical world and/or to support the activities of context-aware services. In such a scenario, traditional approaches based on optimization of routing paths towards a fixed sink [1, 13] (possibly after some limited in-network processing of such data [5, 8]) do not work properly, implying notable energy costs and slow response times..

For sensor networks to become effectively usable as a shared infrastructure for interacting with the physical world, it is necessary to conceive novel approaches to gather information from them in an effective way. The needs of multiple and possibly mobile users must be satisfied quickly and with reasonable accuracy, still ensuring predictable bounds both on the amount of energy globally consumed by the sensor network to satisfy all these needs.

Our approach considers that a sensor network should continuously run, at specific frequency rates and thus with predefined energy costs, a distributed algorithm to identify regions of the network characterized by similar patterns for sensed data. To this end, each node periodically compares the sensed data patterns with its neighbors, so as to eventually self-organize an overlay of logical links partitioning the network into regions. Once region are formed, it is possible to exploit a gossip-based algorithm that – at no additional communication costs – aggregate data on a per-region basis.

The processes of region partitioning and of per-region aggregation, provides multiple and mobile users in need to access information with a prompt access to aggregated values on the local region, and can also acquire a compact perception of the network status.

Although we are aware of a number of current limitations of our approaches, performance studies performed in a simulation environment confirm the effectiveness of our approach and its applicability.

2. Self-organized Region Aggregation

Our approach is based on the following two self-organizing algorithms: *(i)* a distributed algorithm with predictable energy costs is continuously running in the sensor network to partition it into regions characterized by similar patterns of sensed data; *(ii)* The formation of such regions is used to compute, at no additional costs and on a per-region basis, aggregation of sensed data.

2.1. Region Formation

Let us assume to sense, over a certain area covered by a sensor network, a particular property (see Figure 1-a and 1-b), and that each sensor is capable of measuring the local value *v* of such property.

The goal of the region formation algorithm is to have sensors self-organize into disjoint set of regions each characterized by "similar" measures of the property (see Figure 1-c and 1-d). Organization in regions occur via a process of building an overlay of virtual weighted links between neighbor nodes, so that nodes belonging to the same region have strong links, while neighbor nodes belonging to different regions 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 level, while the light level inside a room is always 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.

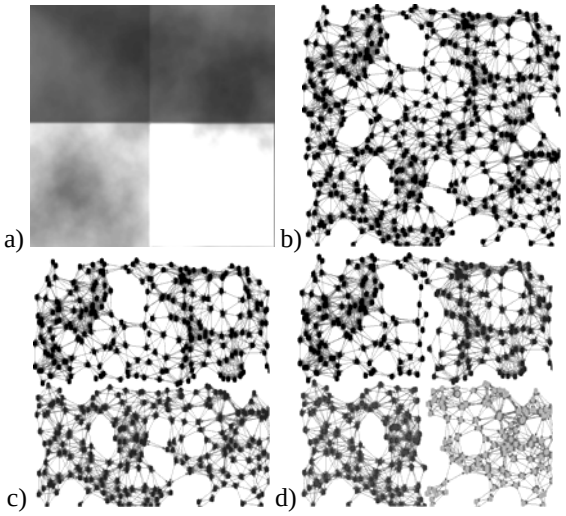


Figure 1. Region self-partitioning. a) a scalar field with 4 regions with different values of a property *v*; b) a sensor network immersed in the above field, with links representing the physical layer; c) overlay defining 2 regions (we show only the logical links); d) overlay defining 4 regions,

Our algorithm works as follows. Let s_i and s_j be two sensors. They can be considered neighbors if they are within the wireless radio range *r*. Let us define the values sensed by s_i and s_j as $v(s_i)$ and $v(s_j)$, and assume that a generic distance function D can be defined for couples of *v* values. Region formation is then based on iteratively computing the value of the logical link $l(s_i,s_j)$ for each and every node of the system, as in the following "*Update_link*" procedure:

```
Update_link:
if D(v(s_i), v(s_j)) < T
        l(s_i,s_j) = min(l(s_i,s_j) + delta, 1)
else
        l(s_i,s_j) = max(l(s_i,s_j) - delta, 0)
```

Where: *T* is a threshold that determines whether the measured values are close enough for $l(s_i,s_j)$ to be re-enforced or, otherwise, weakened, and *delta* is a value affecting the reactiveness of the algorithm in updating link. Details on these parameters follow. What is already clear, though, is that after some iterations, if the $D(v(s_i), v(s_j))$ is lower than threshold *T*, $l(s_i,s_j)$ will converge to *1* otherwise to *0*. In the simplest case, one could consider two nodes s_i and s_j to be in the same region when $l(s_i,s_j)$ is over a threshold T_h, However, to improve stability, we introduced a hysteretic cycle with two threshold T_l and T_h. To run the algorithm, each node stores a vector describing, for each of the neighbors, the current value of the link towards it, and a flag signaling whether the link is connected or not.

The distributed execution of the algorithm is based on a sort of gossip scheme [6]: each node wakes up every *t* time steps, randomly selects a specific number of its neighbors *num_neigh*, exchanges with them the needed data (i.e., the *v* values, plus other data that will be detailed in the following), and then executes the "*Update_link*" procedure for each of the selected neighbors (Due to the broadcast nature of the wireless links in sensor networks, the actual algorithm relies on the receiving neighbors to determine whether they have been selected in the data exchange session or they can just drop the message).

```
Do_forever:
   Wait(t);
   neigh[] = Select_neighbor(num_neigh);
   Foreach(neigh[])
          Data = Exchange_data();
          Update_link(data);
Done
```

Given that, it is rather clear that our algorithm tends to impose a pre-defined, parameterizable, and uniform load, on the system. Each node in the system executes the same amount of operations, the interval *t* determining the frequency of such operations and the number of neighbors *num_neigh* selected at each round determining the communication costs of these operations. Shorter *t* or higher *num_neigh* tend to speed up the convergence of the algorithm, but increase the energy consumed by sensors per time unit.

Let us detail the other parameters of our algorithm: the parameter *delta* determines how fast the link weight *l* changes its value. The choice of this parameter is not crucial, provided that it is chosen small enough to require several cycles of the "*Update_link*" procedure to actually modify the status of link (in other words, it should be notably smaller than the T_l-T_h hysteretic interval). This avoids that random fluctuations of the measured value at a node causes changes in the established regions.

Concerning *T*, a challenging issue in our approach is consists is tackling the difference between the strictly local nature of "*Update_link*" interactions and the inherently global meaning of the threshold *T*. For instance, a difference of 10°C in a wood can be considered relevant during normal days but irrelevant for the sake of fire detection. To deal with this problem,

avoiding the need for a priori information, we opted to define T by exploiting dynamically collected global values of the property *v*. In particular we define *T* as a portion of the whole range of values seen over the network. Using scalar values, we defined *T* as:

$$T = (globalMax - globalMin) * p$$

where *p* is a real number between *0* an *1*. In this way, one can parameterize the sensibility of the algorithm by using a relative value *p* rather than some absolute value requiring a priori knowledge on the range of *v* values. If one wants to obtain very large regions to organize the network based on macroscopic difference one can select *p* close to *1* (*p=0.4*, as in Figure 1-c). If one is interested in more fine-grained region organizations one can select *p* close to *0* (*p=0.05*, as in Figure 1-d). It is worth emphasizing that for each node to locally acquire the *globalMax* and *globalMin* value, one can execute a global aggregation algorithm over the whole network. Simply, as described in [6], each node, when exchanging data with one of its neighbors, can exchange with it the information about the maximum and minimum he knows so far, possibly update its local knowledge, and eventually have the knowledge about the actual *globalMin* and *globalMax* reach each node of the network. Specifically, each node s_i, after having exchanged data with node s_j, executes the following *"Global_aggregation"* procedure:

```
Global_aggregation:
    if(globalMin_i>globalMin_j) globalMin_i=globalMin_j
    if(globalMax_i<globalMax_j) globalMax_i=globalMax_j
```

with $globalMin_i$ and $globalMax_i$ both initialized at v_i.

We emphasize this requires minimal additional effort by nodes. In fact, one can exploit the existing region aggregation noise and its "*Exchange_data*" messages to piggyback the *GlobalMin* and *GlobalMax* values.

2.2. Per-Region Aggregation

Clearly, the local availability of aggregated information over a sensor network may be of some use independently of regions. However, globally aggregated values give very little details on the status of the network, and are definitely of little use for users wishing to acquire information about environmental properties around him/her. For this reason, our approach also exploits per-region aggregation algorithms.

By considering the situation in which regions are already formed, computing aggregation function in a region reduces to executing a gossip-based aggregation algorithm only between those couples of neighbor nodes that are in the same region (i.e., for which the *l* is over the T_h threshold). Again, computing per-region aggregation function does not introduce significant additional burden to the network. The exchange of data between nodes can occur by piggybacking over the existing messages, and the computation of local aggregation algorithms reduces to adding a simple "*Local_aggregation*" function in the main body of our basic scheme, as follows:

```
Do_forerever:
    Wait(t);
    neigh[] = Select_neighbor(num_neigh);
    Foreach(neigh[])
            Data = Exchange_data();
            Update_link(data);
            Global_aggregation();
            If(connected) Local_aggregation();
Done
```

The "*Local_aggregation*" function can include the identification of the local minimum and the local maximum of some sensed value *w* within the region, or the calculus of the average *Avg* of some value *w*. Currently, in our scheme, we also decided to enforce two peculiar aggregation functions that are of great use for facilitating the gathering of information by users.

- The first aggregation function 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 a "hop counter" initialized to *0*. By having such counter re-propagated by each node on per-minimum basis, the results is that each nodes in the region eventually becomes aware of its distance form the closest border. This is use for enabling each node to locally estimate the "radius" within which the aggregated data are definitely meaningful, and to identify whether or not it can properly answer to a query.
- A second aggregation function 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. This can be very useful for mobile users to identify that they are changing regions, as well as to enable a quick and compact identification of all the regions within an area.

It can be shown that gossip-based aggregation processed does not experience problems if executed on a growing number of nodes, as in the region formation transitory.

This also applies for the identification of the region leader (when two regions merge, one of the two leaders will eventually recognize it lost his role). Similar considerations apply to the case in which new sensors are dynamically added in the system.

Let us now consider the case in which some existing regions shrinks, either because a confining region has expanded or because some sensor nodes have died. In this case, two problems arise:

- the values computed by the local aggregation functions may no longer be valid (e.g., the former maximum may have left the region) but they will not be properly updated, due the cumulative nature of gossip-based aggregation;
- the region leader may have exited the region.

To overcome the former problem, we decided to enforce a sort of "evaporation" of the values computed by the local aggregation algorithms. The local aggregated values in a node are slowly moved towards the values sensed by 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 maximum of a region, and assume that each node in a region has already locally available the value of such maximum. Now, have each node slightly "evaporate" such value by making it approach the local value. If the node holding such maximum is still in the region, a node will receive its value undoing the evaporation effects. If the node holding the maximum, instead, has exited the region, evaporation will enable to stabilize the new maximum at each node, after proper evaporation. Similar considerations apply, e.g., to the calculus of the average.

To overcome the region leader problem, each node keeps track of the "oldness" of the value of the leader *ID* (accounting for the number of cycles of the algorithm since the last time it received from some node such *ID*). Whenever such oldness becomes excessive, the current leader *ID* is considered obsolete and a new leader is identified and elected. Clearly, all the above solutions also help to deal with sensor networks immersed in environment with dynamically changing properties, and make our approach self-organizing and self-adaptable.

3. Performance Evaluation

We have performed numerous experiments based on simulations. First, we wanted to evaluate the effectiveness of the region detection algorithm. Second, we wanted to evaluate the convergence and accuracy level of the aggregation algorithms. The results of the simulations were obtained by simulating scalar fields in which the sensor network is immersed. Though we have conducted several experiments on fields with different shapes and values, we have always obtained comparable results from both qualitative and quantitative viewpoint. Therefore, we report here on an environment filled with 500 wireless sensors disposed over a random graph such that the mean number of neighbors for each node is 15 (similarly to the sensor network of Figure 1-b). The simulated scalar field exhibits values v such that four different quadrants are recognizable (as in Figure 1-a). Each quadrant has a fixed mean m and variance s. Starting from the top left quadrant and proceeding clockwise, they could be identified as q_1, q_2, q_3, q_4. Mean values $m_1..m_4$ are respectively 120, 80, 20, -20. Variances $s_1..s_4$ are arranged such that in each quadrant are allowed values v in range $[m - 2, m + 2]$.

3.1. Region Formation

From a static viewpoint, we can analyze, independently from the speed of convergence, which are stable states reached by the network and evaluate the effects of related parameter p. As described in Subsection 2.1 and as shown in Figure 1, variations on the parameter p induce the network to self-partition into regions of different sizes.

From a dynamic viewpoint, we can show how variations of the gossip percent *num_neigh* and the sleep cycle t affect the speed of convergence and the accuracy of the algorithm. Initially all nodes are not connected with any neighbor. We collect data over the first *255* cycles. Within cycles from *0* to *128* p is set to *0.4*. During this interval the network converge to a status similar to that of Figure 1-c, i.e., splitting the network into regions. At cycle *129*, we changed p from *0.4* to *1.0* , making the network re-compact into a single region (as in Figure 1-b). Figure 2 shows the evolution in the average number of nodes per region as time passes, by varying the gossip percentage. Values are collected at the completion of each simulation cycle, and figure shows that the number of nodes of the region start from *0*, grow to *250* during the first phase *[0 – 128 cycles]* and than reaches 500 during the second phase *[129 – 255 cycles]*.

Clearly, reducing the gossip percentage or increasing the sleep period t makes the network slower in the region detection process. It is trivial to understand that any increase in t corresponds to a directly proportional increase in the time required for region detection. The impact of decreasing *num_neigh* is more interesting. In fact, as Figure 2 shows, lowering *num_neigh* only slightly slow-down the convergence time.

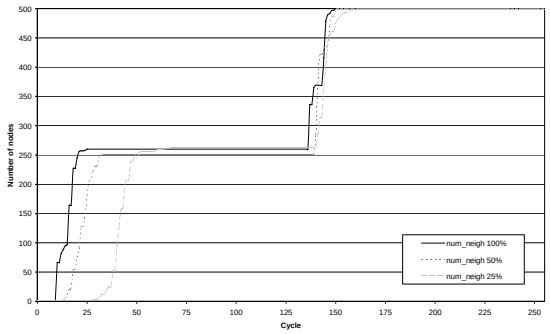
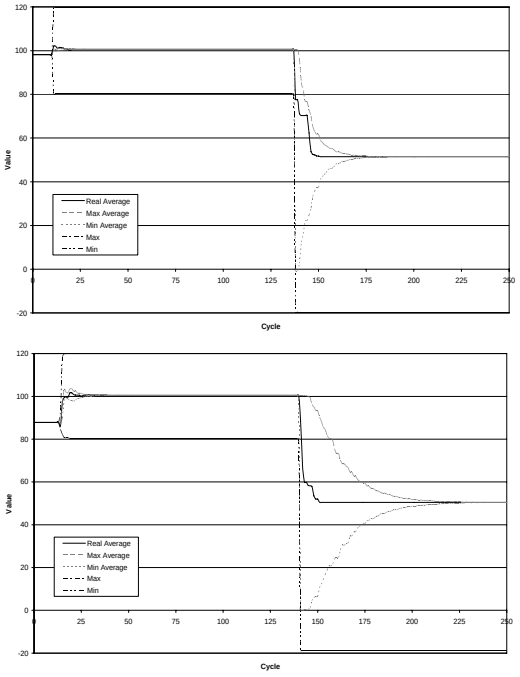


Figure 2. Evolution of region detection for *t = 1* and for various values of *num_neigh.*

3.2. Per-Region Aggregation

Let us focus on the behavior of the RAN approach in evaluating aggregated values. From the static viewpoint, all local aggregation algorithms experiences correctly reach convergence towards the corrent (real) value. From the dynamic viewpoint, Figure 3 shows the trend of several values aggregated on a per region basis. 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. Figure 3-a show results obtained with *num_neigh=1.0* and *t=1*. Figure 3-b shows results obtained reducing *num_neigh* to *0.5*. Clearly, reducing the gossip percentage expressed by *num_neigh* or increasing the sleep period *t* make the network slower not only in region detection but also in correctly evaluating locally aggregated values. Since, as already discussed, the impact of *t* is of direct proportionality, we only focus here on the impact of *num_neigh*. All the graphs in Figure 3 show the same trend. During the first cycles while links are being reinforced, all the aggregated values don't change. At the beginning (cycle *0*), when the region starts forming is clearly visible a fast convergence of the local maximum and minimum to their new values respectively of *120* and *80*. Average related values have a relatively small transitory and eventually reach the value of *100* as expected. At cycle 128, *p* is changed to *p* = 1.0 and 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 slowly converge to the expected value of *50*. Observing Figure 3 is clear that different aggregate values behave differently varying *num_neigh*. In particular accuracy of average related values are really more sensible to variations of *num_neigh* than the local minimum and maximum have.

Summarizing, there is a clear trade-off between energy consumption and accuracy: higher *num_neigh* and the lower *t* clearly provides for more accuracy over time, but overall increase the energy consumed. In any case, to decrease energy consumption without compromising too much accuracy, it is worth exploiting both an appropriate increase of *t* complemented by an appropriate lowering of *num_neigh*.



a)

b)

Figure 3. Per region aggregated values. Minimum estimate of the maximum, maximum estimate of the minimum, minimum and maximum estimates of the average and real value of the average. a) *num_neigh = 1.0, t* = 1; b) *num_neigh = 0.5, t* = 1.

4. Related Work

Most work on data gathering and aggregation in sensor networks assumes the presence of fixed sinks (base stations) to which sensed data flow. The basic approach is that of having sensors build a tree rooted at the sink and supporting the routing of sensed data towards it [13]. Some form of in-network data aggregation (e.g., averaging) can be performed as data from sensors climbs the tree [5, 8], and various optimization can apply in tree formation [1]. In any case, these approaches can hardly

apply for shared infrastructural sensor networks, because the costs of building a tree on demand for many possible users at different and varying locations would be high, both in terms of energy and response time.

Several research works in the area of sensor networks start recognizing the need to promote direct access to sensor data by multiple and mobile users [4, 11]. These works mostly focuses of defining suitable general-purpose primitives and language constructs to enable users to flexibly query the network and obtain information about individual sensor data and aggregated data related to specific regions. However, apart from a few exceptions [9], none of these systems faces the problem of how to implement the query functionalities, i.e., of what specific data gathering and aggregation algorithms should run in the sensor network.

Some in-network algorithms for self-organization of region partitioning in sensor networks have been proposed [2, 12], sharing some basic principle with our approach. The key differences being that: *(i)* these algorithms require a priori information about the typically patterns exhibited by the environment, while our approach does not and it is fully self-organizing; *(ii)* these algorithms are not conceived for other goals than recognizing regions, while our approach goes further, by exploiting region partitioning for computing aggregation and for supporting efficient queries by multiple and mobile users.

5. Conclusions and Future Work

The proposed approach enables a sensor network to analyze the patterns of sensed information so as to self-partition into regions characterized by similar sensing patterns, and then to aggregate data on a per-region basis. In this way, multiple and mobile users can extract meaningful information from the network at very limited costs and with notable accuracy.

We are currently working to overcome some limitations of our approach. First, we would like it to support multiple overlays and general-purpose queries, there included inter-region global queries. Second, we would like it to work even without the availability of global aggregated information. Last but not least, we are in the process of verifying the effectiveness of the approach on a real sensor network testbed.

Acknowledgments. Work supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

References

[1] Athanassios Boulis, Saurabh Ganerival, and Mani B. Srivastava, “Aggregation in sensor network: An Energy-Accuracy Trade-off”, IEEE SANPA, 2003.

[2] E. Catterall, K. Van Laerhoven and M. Strohbach. "Self-Organization in Ad-Hoc Sensor Networks: An Empirical Study". The 8th International Conference on the Simulation and Synthesis of Living Systems, Sydney, (AU). MIT Press, pp. 260-264.

[3] C.-Y. Chong, S. P. Kumar, “Sensor networks: Evolution, opportunities, and challenges”, Proceedings of the IEEE, 91(8):1247-1256, Aug. 2003.

[4] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. Murphy, G. P. Picco, "Mobile Data Collection in Sensor Networks: The TinyLime Middleware", Journal of Pervasive and Mobile Computing, (4)1:446-469, 2005.

[5] Johannes Gehrke, Samuel Madden, “Query Processing In Sensor Networks”, IEEE Pervasive Computing, 3(1):46-55, Jan. 2004.

[6] M. Jelasity, A. Montresor, O. Babaoglu, “Gossip-based Aggregation in Large Dynamic Networks”, ACM Transactions on Computer Systems, 23(3):219-252, 2005.

[7] Chenyang Lu, Guoliang Xing, Octav Chipara, Chien-Liang Fok, Sangeeta Bhattacharya: “A Spatiotemporal Query Service for Mobile Users in Sensor Networks”, 25th International Conference on Distributed Computing Systems, June 2005, pp. 381-390.

[8] Samuel Madden, Joseph M. Hellerstein, “Distributing queries over low-power wireless sensor networks”, ACM SIGMOD International Conference on Management of Data, 2002.

[9] G. Mottola, G. P. Picco, "Logical Neighborhoods: A Programming Abstraction for Wireless Sensor Networks", Proceedings of the 2nd International Conference on Distributed Computing in Sensor Systems, San Francisco (CA), June 2006.

[10] R. Müller, G. Alonso, “Shared Queries in Sensor Networks for Multi-User Support”, Technical Report 508, ETH Zürich, Institute of Pervasive Computing, Feb. 2006. 23(3):219-252, Aug. 2005.

[11] R. Newton, M. Welsh, “Region Streams: Functional Macroprogramming for Sensor Networks”, Proceedings of the 1st International Workshop on Data Management for Sensor Networks, Toronto (CA), pp. 78 – 87, 2004.

[12] A. Panangadan. G. S. Sukhatme, “Data Segmentation for Region Detection in a Sensor Network”, Center for Robotics and Embedded Systems, University of Southern California, Technical Report 05-005, 2005.

[13] J. Polastre, R. Szewcyk, A. Mainwaring, D. Culler, J. Anderson, "Analysis of Wireless Sensor Networks for Habitat Monitoring", in Wireless Sensor Networks, Kluwer Academic Publishers (NY), 2004, pp. 399-423.