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 analyze, in a distributed way and with predefined energy costs, the patterns of sensed information so as to self-partition itself into spatial regions of nodes characterized by similar patterns. Such regions can then be used to aggregate data on a per-region basis and to enable multiple and mobile users to extract meaningful information at very limited and pre-defined costs.

1. Introduction

Sensor network technologies are gaining increasing interest and increasing diffusion [Est02, ChoK03]. In this area, most of current researches as well as most of systems deployed so far consider that sensor networks are to be designed and deployed with the only goal of monitoring some specific physical phenomena and of reporting data back to some fixed base station acting as a sink [Pol04, BouG03], possibly after some limited in-network processing of such data [MadH02, GehM04].

However, the trend toward ubiquitous mass deployment of sensor networks will make them become more and more as a general shared infrastructure for the use of multiple [MulA06, Cur05] and possibly mobile users [Cur05, Lu05]. The idea is that users in an environment can 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, approaches based on optimization of routing paths towards a fixed sink do now 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, ensuring reasonable bounds both on the amount of energy consumed to satisfy the need of several possibly mobile users and on the response time and accuracy of data provided to them.

The idea underlying our approach is that of having the network 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 with its neighbors the sensed data patterns. A logical link between nodes is re-enforced in the case of relevant similarity and weakened in the case of relevant dissimilarity. Eventually, the execution of the algorithm results in a self-organized overlay of logical links partitioning the network in regions.

Upon formation of such regions, aggregation of data can act on a per-region basis, by exploiting a gossip-based algorithm that, by relying on piggybacking, requires no additional communication costs. Due to this in-network process of per-region aggregation, which also avoids the accuracy losses of global aggregation algorithms, multiple and mobile users in need to access information can be provided, with very limited costs, prompt access to aggregate values on the local region, and can also acquire a compact perception of the overall status of the network.

2. Region Aggregation Noise

The "Region Aggregation Noise" (RAN) approach considers the following: *(i)* a distributed algorithm is running in the network as a sort of "background noise" with a predictable energy cost to partition the sensor network into regions characterized by similar patterns for 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, so that users accessing the network for gathering information can be provided with such pre-computed aggregated data at limited costs.

2.1. Region Formation

Let us assume to sense, over a certain area covered by a sensor network, a particular property (see e.g. Figure 1-a and 1-b), and that each sensor is capable of measuring the local value *v* of such property. For instance, *v* could be a temperature, or a light level, or whatever property a sensor is capable to infer about its sensed portion of the environment.

The goal of the region formation algorithms is to have sensors self-organize into disjoint set of regions each characterized by "similar" measures of the property (e.g.. see Figure 1-c and 1-d). Organization is regions occur via a process of building an overlay of virtual weighted links between neighbor nodes, such 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.

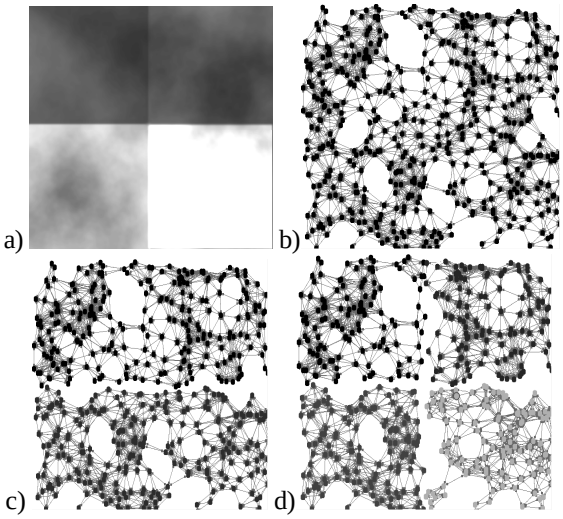


Figure 1. Region self-partitioning. a) a scalar field with 4 regions with different values of a property *v*; b) a sensor network immersed in the above field, with links representing the physical layer; c) overlay region organization with p=0,4 defining 2 regions (we show only the logical links); d) overlay region organization with p=0,05 defining 4 regions,

The problem of clustering the network into regions with similar properties is very well studied and several centralized algorithms have been proposed to this purpose nowadays (assuming the availability of sensor values), but it is more challenging to perform in a strictly local and distributed fashion.

Basically, our algorithm work 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. Define the values sensed by s_i and s_j as $v(s_i)$ and $v(s_j)$, and let us assume that a generic distance function D can be define for couples of v values (thus defining v as a metric space), i.e., $D(v(s_i), v(s_j))$. Region formation is then based on interatively 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. Transitively, two nodes s_h and s_k are defined in the same region if and only if exists a chain of nodes such that each pair of neighbors in the chain are in the same region. For the actual execution of the algorithm, each node stores a vector describing, for each of the neighbors, the current value of the link towards it and a flag (also necessary for hysteresis) signaling the status of the link (connected or not).

The distributed execution of the algorithm is based on a sort of gossip scheme [JelMB05]: each node periodically wakes up, randomly selects a specific number (or a specific percentage) of its neighbors, exchange with them the needed data (i.e., the *v* values, plus other data that will be detailed in the following), and then execute the "*Update_link*" procedure for each of

the selected neighbors:

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

At this point, 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 execute the same amount of operations, the interval *t* determining the frequency of such operations and the number of neighbor *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 sensor per time unit. What matters, is that one can select the proper trade-off between convergence times and accuracy and energy cost, an issue that will also be analyzed in the performance evaluation section. In other word, one can select the "degree of noise" of our algorithm and, so, the energy consumed over time.

Let us now go into more details about the other parameters of our algorithm.

Concerning the parameter *delta*, it 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 or temporary fluctuations of the measured value at a node continuously 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*. How can two nodes evaluate which is the right threshold to establish if they are similar enough to be in same region or not if they don't know anything about the rest of the network ? 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* (as in Figure 1-c). If one is interested in more fine-grained region organizations one can select *p* close to *0* (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 [JelMB05], each node, when exchanging data with one of its neighbors, can exchange with it the information about the maximum and minimum he know 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 very minimal additional effort by nodes. In fact, one can exploit the existing region aggregation noise and its "*Exchange_data*" messages to exchange the *GlobalMin* and *GlobalMax* values, by piggybacking with such messages the additional data needed, and then computing the "*Global_aggregate*" function after the "*Update_link*" procedure inside the main algorithm body. Moreover, one can also decide to exploit the same schema to compute any additional distributed aggregation algorithms (e.g. computing the average), and possibly even to compute aggregations over properties different from *v*.

2.2. Gossip-based 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 info 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 (the handling of transitory situations will be discussed later on), 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 (computed as in the global case), or the calculus of the average *Avg* of some value w. In this case, the local aggregation for a node s_i, after having exchanged data with connected node s_j, simply works as follows:

$$Avg_i(w) = (Agv_i(w)+Avg_j(w))/2$$

with $Avg_i(w)$ simply initialized at the local value w_i.

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 frontiers 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 by 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 of the frontier. 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. We also plan to experience more sophisticated aggregation function to enable nodes to locally reach a higher understanding of the shape and topology of the local region, possibly relying on existing work of distributed topology recognition.

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 (and the leader itself will recognize itself as that). 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.

Let us now analyze the dynamic behavior of the system during region formation and region re-shaping (changes in the values of the property v upon which region formation relies can induce changes in the shape and dimensions of regions).

In general, the initial values of the virtual links l between nodes are irrelevant for region formation. Therefore, let us assume an initial situation in which all nodes are disconnected from each other (i.e., each node is a region in itself). As the algorithm will start running, nodes with similar values of v will start connecting with each other, and sets of regions with growing dimensions will start forming and possibly merge each other, until a stable situation will be reached.

Concurrently with the above region formation process, the local aggregation procedure start executing as soon as two nodes gets virtually connected in the same region, and the computing of aggregated data gradually involves more and more regions, eventually converging when a stable region situation is reached. It can be shown (and it is quite intuitive indeed, due to the cumulative nature of aggregation) 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 apply for the identification of the region leader (when two regions merge, one of the two leaders will eventually recognize it is no longer such).

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: (i) the values computed by the local aggregation functions may no longer be valid (e.g., the former maximum may have left the region) but – due the cumulative nature of gossip-based aggregation – will not be properly updated; (ii) 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 (except for the leader election algorithm). In other words, the local aggregated values at a node are slowly (compared to the convergence time of the aggregation algorithms) moved towards the 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 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 diminish approach the local value. If the node holding such maximum is already in the region, a node will be made aware of this soon (i.e., since evaporation is slow, before the node itself has "evaporated" the value too much) and can undo 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.

The second problem is somewhat similar, but cannot be tackled by evaporation (the leader *ID* is not a value that can be tolerate approximation). Accordingly, the solution is inspired by the same principle, but is somewhat less elegant and fluid. 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 (i.e., the new node with the minimal *ID*) is identified and elected.

Clearly, all the above solutions also help to deal with sensor networks immersed in environment with dynamically changing properties, and overall make the RAN approach fully self-organizing and self-adaptable.

3. Evaluation

We have performed numerous experiments based on simulations using the Repast framework [Repast06]. Our goal was twofold. 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, and the trade-off between accuracy and energy consumption.

The results of the simulations were obtained by simulating scalar fields in which the sensor network is immersed similar to that of Figure 1. 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 (i.e., qualitatively assimilable 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 $m_1..m_4$ of f in $q_1..q_4$ are respectively 120, 80, 20, -20. Variances $s_1..s_4$ are arranged such that in each quadrant are allowed values v in range $[m - 2, m + 2]$.

Network behaviour can be described from both a static and dynamic point of view. From the former 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. From the latter we show the dynamic behavior of the network, the speed of convergence and the accuracy level depending on *num_neigh* and t.

3.1. Region Detection

From a static viewpoint, as described in Subsection 2.1 and as shown in Figure 1, variations on the parameter p induce the network in self-partition into regions of different sizes. The same behavior has been verified to apply for networks immersed in fields with different shapes and with different sizes.

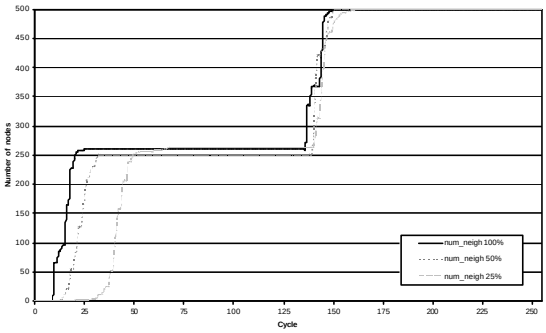
Let us now switch to the dynamic viewpoint and show how variations of the gossip percent *num_neigh* and the sleep cycle t affect the speed of convergence and the accuracy of the region detection algorithm. Let's consider a simulated a 500-nodes sensor network and a scalar field similar to that of Figure 1. 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).

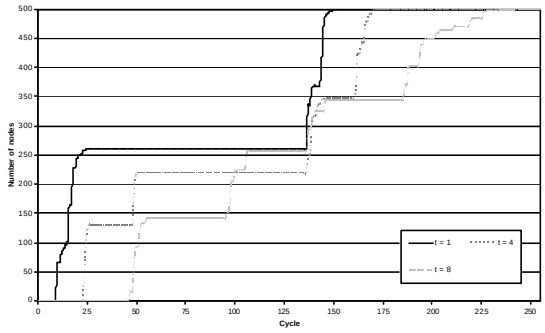
In Figure 2-a we show the evolution in the average number of nodes per region as time passes, by varying the gossip percentage. Figure 3-b shows the same kind of evolution but by varying the sleep period *t* of sensor nodes. Values are collected at the completion of each simulation cycle. Both the graph show 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* make the network slower in the region detection process.

From Figure 2, it also emerges that the speed of the network is less influenced by variations of *num_neigh* than by variations of *t*.

The strange "stairs-like" trend of data lines obtained by setting *t=4* and *t=8* (Figure 2-b) clearly show the non-linear nature of the algorithm. These are mostly due to the fact that, when a region is forming, lots of sub regions are growing within it connecting the most similar neighbors. Only when the new actual minimum ID of the new region reaches a node, such node recognize it is becoming part of a new region.



a)



b)

Figure 2. Evolution of region detection. a) *t = 1*. *num_neigh = 1*, *num_neigh = 0.5*, *num_neigh = 0.25*; b) *num_neigh = 1*. *t = 1*, *t = 4*, *t = 8*.

3.2. Local Aggregation

Let us know focus on the behavior of the RAN approach in evaluating aggregated values.

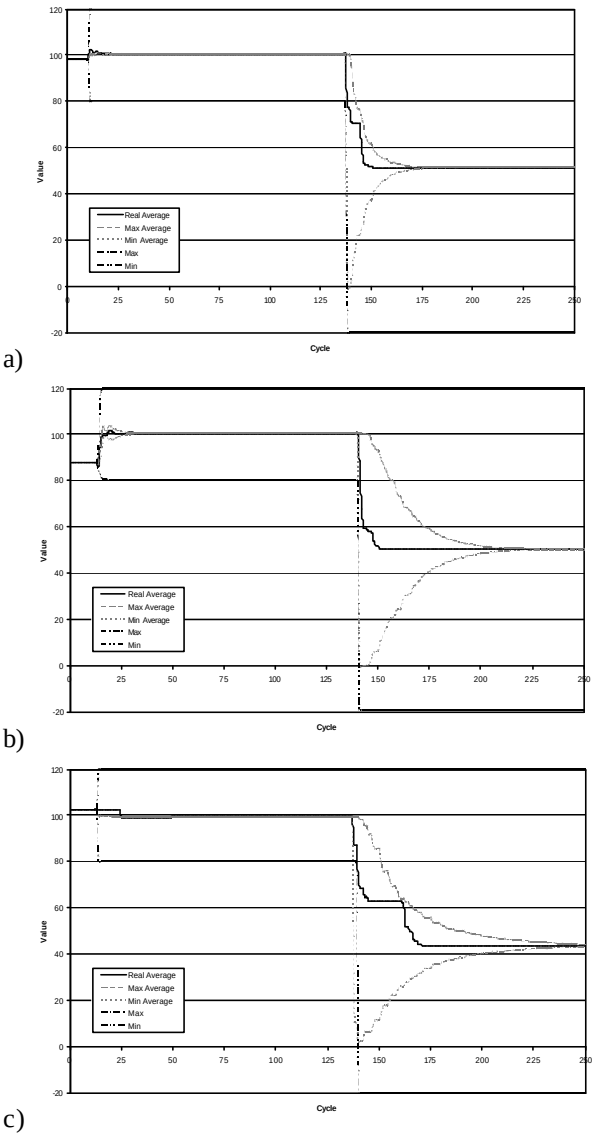
From the static viewpoint, all local aggregation algorithms experiences corrently reach convergence towards the corrent (real) value.

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 4-a show results obtained with *num_neigh=1.0* and *t* = 1. Figure 3-b and 3-c show results obtained reducing *num_neigh* to *0.5* and increasing *t* to *4*, respectively, Clearly, reducing the gossip percentage or increasing the sleep period *t* make the network slower not only in region detection but also in correctly evaluating locally aggregated values.

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* and *t*. In particular accuracy of average related values are really more sensible to variations of *num_neigh* and *t* than the local minimum and maximum have.

To summarize this, 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. Due to the high convergence speed of *Max* e *Min* showed under all conditions tested and to the fact that regions are expected to have relatively limited size, scalability of

the RAN approach should not be a major issue. We tested it with sensor networks up to 10000 nodes obtaining similar results.



a)

b)

c)

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; c) *num_neigh* = *1.0*, *t* = 4.

4. Application Areas and Current Limitations

4.1. Application Areas

The most direct and general-purpose way of exploiting the RAN approach – and the one that indeed motivated our work – is for supporting queries by multiple and mobile users. A user in an environment that wants to retrieve information about the surrounding will typically access the nearest sensor and query it about some local patterns of sensed data, e.g. "give me the average value of the temperature in this room" or "give me the maximum value of temperature within 500 meters". At this point, if the sensor network has already provided for aggregating such data on a per region basis and the queries relate to the local region (which the node itself can recognize by estimating the distance of the closest border via the local hop counter), the sensor can immediately answer the query without further burdening the sensor network with computation and communications. The limitations in supporting more general and more global queries are analyzed later on in this section.

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 recognize unusual patterns of sensing and, in case, to automatically generate alarms. For example, we are currently cooperating with the geological department of the Modena Apennine to exploit our approach for effective landslide detection. Other examples in this direction include the possibility of detecting anomalies in buildings, streets, or parks.

More in general, 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. In-network aggregation will become the only solution to extract useful information from them. Accordingly, approaches such as RAN, which enables the sensor network to self-organize regions of aggregation and to report at limited cost concise information about such regions is likely to become increasingly important.

Last but not least, the RAN approach can be seen as a way to effectively extract high-level semantic knowledge about the structure and characteristics of an

unknown environment, for use by context-aware and location-based services [Bau06].

4.2. Limitations

A shortcoming of the RAN approach is in its limited support for general queries. In fact, one has to a priori identify what type of sensed data (e.g., the temperature) or data function (e.g., some combination of temperature and light) to exploit as the basis for identification of regions and what data to exploit for the subsequent per region aggregation. However, if sensor networks are to become a shared infrastructure for the use by multiple mobile users, it may be also expected that different users will need to access different types information among the several that a sensor network can provide.

To overcome this problem, it is possible to think at exploiting our background algorithms for the concurrent building of several virtual overlay region partitions, each corresponding to different kinds of sensed data or data functions. Moreover, one could think at the possibility of dynamically "injecting" into a network the specification of any particular data function and/or aggregation function. This would enable to have the network dynamically start building an additional overlay region partitioning based on such data, or computing the newly specified aggregation functions over the existing region partitions, or both. This would open up the possibility of supporting general region-based queries, as e.g. proposed by Region Streams [NewW04] and Logical Neighborhoods [MotP06]. Although enforcing multiple partitions and multiple aggregation functions would not require additional messages between sensor nodes, but only the piggybacking of additional information in existing messages, the costs of building and maintaining multiple virtual overlays and several aggregation functions have to be carefully evaluated.

Another, less critical, limitation of RAN is that it is able to effectively answer queries within a region, but fall short in providing users information about what it happening outside the region. For instance, if a user asks "the average temperature within 500 meter" and the query is performed at a distance of 300 meters from the closest confining regions, the sensor will not be able to answer immediately. To overcome this problem, we are planning to implement efficient and low-costs inter-region aggregation algorithms based on gossiping.

Finally, the region partitioning algorithm we have experience so far requires, at each node, the availability of global data representing the maximum and the minimum of the sensed data pattern to properly identify regions. This represents indeed a limitation from a scalability point of view. On this base, we are currently experiencing with a modified algorithms capable of identifying regions on the basis of local information only.

5. 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 [Pol04]. Some form of in-network data aggregation (e.g., averaging) can be performed as data from sensors climb the tree [MadH02, GehM04], and various optimization can apply in tree formation [BouG03]. In any case, thee 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 unbearable, 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 [NewW04, Cur05]. 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 [MotP06], 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.

To idea of exploiting aggregation algorithm continuously running in the network so as to provide locally to each node a more global picture of the of the sensed environment has been originally proposed for P2P networks [JelMB05] and, later, also for [DimSW06]. However, for sensor networks, the general approach of gossip algorithms executing over the whole network is not satisfactory: users querying a network to be interested in aggregated values related to a local region (as in RAN) rather than in global values related to the whole network.

Some in-network algorithms for self-organization of region partitioning in sensor networks have been proposed [CatWS02, PanS05], sharing some basic principle with our RAN approach. The key differences

with RAN are that: *(i)* these algorithms require a priori information about the typically patterns exhibited by the environment, while RAN does not and it is fully self-organizing; *(ii)* these algorithms are not conceived for other goals than recognizing regions, while RAN goes further, by exploiting region partitioning for computing aggregation and for supporting efficient queries by multiple and mobile users.

6. Conclusions and Future Works

If sensor networks are going to become a pervasive shared infrastructure, algorithms and tools will be required to support querying by multiple and mobile users other than by fixed sinks. The proposed approach enables a sensor network to analyze the patterns of sensed information so as to self-partition the network 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 extend our approach to: support multiple overlays and general-purpose queries; support inter-region global queries; 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 testbeds, other than in a simulated environment.

References

[BouG03] Athanassios Boulis, Saurabh Ganerival, and Mani B. Srivastava, “Aggregation in sensor network: An Energy-Accuracy Trade-off”, Proceedings of IEEE SANPA, May 2003.

[CatWS02] 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.

[ChoK03] C.-Y. Chong, S. P. Kumar, “Sensor networks: Evolution, opportunities, and challenges”, Proceedings of the IEEE, 91(8):1247-1256, Aug. 2003.

[Cur05] 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, Dec. 2005.

[DimSW06] A. G. Dimakis, A. D. Sarwate, M. J. Wainwright, “Geographic Gossip: Efficient Aggregation for Sensor Networks”, Proceedings of the International Conference on Information Processing in Sensor Networks, Nashville (TN), ACM Press, April 2006.

[GehM04] Johannes Gehrke, Samuel Madden, “Query Processing In Sensor Networks”, IEEE Pervasive Computer Journal, April 2004.

[JelMB05] M. Jelasity, A. Montresor, O. Babaoglu, “Gossip-based Aggregation in Large Dynamic Networks”, ACM Transactions on Computer Systems.

[Lu05] 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.

[MadH02] Samuel Madden, Joseph M. Hellerstein, “Distributing queries over low-power wireless sensor networks”, Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, June 2002.

[MotP06] 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.

[MulA06] 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.

[NewW04] R. Newton, M. Welsh, “Region Streams: Functional Macroprogramming for Sensor Networks”, Proceedings of the 1st International VLDB Workshop on Data Management for Sensor Networks, Toronto (CA), pp. 78 – 87, 2004.

[PanS05] 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.

[Pol04] 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.