Self-organization and Local Learning Methods for Improving the Applicability and Efficiency of Data-Centric Sensor Networks*

Gabriele Monti and Gianluca Moro

DEIS - University of Bologna,
Via Venezia, 52,
I-47023, Cesena (FC), Italy
{gabriele.monti4,gianluca.moro}@unibo.it

Abstract. In data-centric sensor networks each device is like a minimal computer with cpu and memory able to sense, manage and transmit data performing in-network processing by means of insertions, querying and multi-hop routings. Saving energy is one of the most important goals, therefore radio transmissions, which are the most expensive operations, should be limited by optimizing the number of routings. Moreover the network traffic should be balanced among nodes in order to avoid premature discharge of some devices and then network partitions. In this paper we present a fully decentralized infrastructure able to self-organize fully functional data centric sensor networks from local interactions and learning among devices. Differently from existing solutions, our proposal does not require complex devices that need global information or external help from systems, such as the Global Positioning System (GPS), which works only outdoor with a precision and an efficacy both limited by weather conditions and obstacles. Our solution can be applied to a wider number of scenarios, including mesh networks and wireless community networks. The local learning occurs by exploiting implicit cost-free overhearing at sensors. The work reports an extensive number of comparative experiments, using several distributions of sensors and data, with a well-know competitor solution in literature, showing that an approach fully based on self-organization is more efficient than traditional solutions depending on GPS.

1 Introduction

Self-organization is becoming a promising paradigm to cope with complex systems and to reduce their costs, in fact it leads, in general, to properties like the self-configuration, self-administration and self-healing etc., namely to systems that work without, or drastically limiting, the human interventions. Application scenarios where it can be inconvenient or unfeasible to set up a system by

* Work partially funded by the european project DORII: Deployment of Remote Instrumentation Infrastructure Grant agreement no. 213110.

N. Bartolini et al. (Eds.): QShine/AAA-IDEA 2009, LNICST 22, pp. 627–643, 2009.

configuring or implementing every single component are also emerging in large wireless ad-hoc networks.

Examples of ad-hoc networks are mesh networks, sensors networks and wireless community networks (i.e. static or quasi-static networks), while vehicular networks are an interesting example of MANET (mobile ad-hoc networks). Compared to wired networks, wireless networks have unique characteristics. In wireless networks, node failures may cause frequent network topology changes, which instead are rare in wired networks. In contrast to the stable link capacity of wired networks, wireless link capacity continually varies because of the impacts from transmission power, receiver sensitivity and interference. Additionally, wireless sensor networks have strong power restrictions and bandwidth limitations.

In this paper we focus on self-organizing sensor networks, though our work is easily suitable for all static (or quasi static) wireless ad-hoc networks, as we highlight in some rows. Several kinds of sensor applications have been developed in recent years. In some applications, a large volumes of data or events are continuously collected, aggregated/synthesized and stored by sensors for in-network processing. Data-Centric Storage (DCS) scheme emerged from the sensor network literature as the most efficient one for storing and processing data directly within a sensor network. This kind of networks are possible thanks to a new generation of sensors equipped with memory for storing data and processors for executing moderately demanding algorithms. In other words such sensors are similar to minimal computers but with more restrictions, hence the solution presented in this paper is also suitable to the mentioned above networks composed by more powerful devices.

In DCS, events are placed according to their event types, which refers to predefined attribute's values (temperature and pressure, for instance). Hence data or events can be named by attributes and logically represented as relations in distributed databases [7] [1] [4].

In this paper we present a new solution based on a local learning method that improves the performance of W-Grid infrastructure [10] [11] [9] [8], both in terms of routings up to 12% and of traffic balancing, without affecting energy consumptions; we highlight that it is difficult to gain both performance on routing and on balancing just because there is a trade-off between them. The solution does not require GPS because each device receives a virtual coordinate reflecting its local connectivity with other neighbour devices and each of them uses this information to perform routings and to forward exact match query, namely a query to search a single exact data. This means a greater applicability than existing solution based on GPS. The work also present how the infrastructure manages range queries, which are more complex searches involving two or more attributes/dimensions. The in-network data management occurs spontaneously by observing that each device receives a set of multiple unique virtual coordinates, each of which represents also a portion of the data indexing space for which a device is assigned the management responsibility.

Section 2 describes the related works, in Section 3 we briefly present the main features of the infrastructure. Section 4 illustrates the management of

data and queries. Section 5 introduces the cost-free local learning capability. Section 6 describes an extensive number of comparative experiments according to several scenarios; the performances are compared with the preceding version of W-Grid and with a well-known competitor solution in literature that requires GPS. Section 7 reports final considerations.

2 Related Works

Routing is necessary whenever a data sensed (generated) must be transmitted elsewhere in the network, including an external machine, proactively or reactively according to periodic tasks or queries submitted to the network system.

As stated before, we do not consider sensor networks which simply transmit data externally at a remote base station, we focus on advances wireless sensor networks in which data or events are kept at sensors, are indexed by attributes and represented as relations in a virtual distributed database. For instance in [4,15], data generated at a node is assumed to be stored at the same node, and queries are either flooded throughout the network [4].

In a GHT [13], data is hashed by name to a location within the network, enabling highly efficient rendezvous. GHTs are built upon the GPSR [5] protocol and leverage some interesting properties of that protocol, such as the ability to route to a sensors nearest to a given location, together with some of its limits, such as the risk of dead ends. Dead end problems, especially under low density environment or scenarios with obstacles or holes, are caused by the inherent greedy nature of the algorithm that can lead to situation in which a packet gets stuck at a local optimal sensors that appears closer to the destination than any of its known neighbors. In order to solve this flaw, correction methods such as perimeter routing, that tries to exploit the right hand rule, have been implemented. However, some packet losses still remain and furthermore using perimeter routing causes loss of efficiency both in terms of average path length and of energy consumption. Besides, another limitation of geographic routing is that it needs sensors to know their physical position adding localization costs to the system. In DIFS [3], Greenstein et al. have designed a spatially distributed index to facilitate range searches over attributes.

Our solution is more similar to the multi-dimensional distributed indexing method for sensor networks developed in [6] and [14], but differently from our approach they require nodes to be aware of their physical location and of network perimeter; moreover they employ GPSR for the physical routing. GPSR routing performances are heavily affected by network topology (e.g nodes density or obstacles) and it cannot work in indoor environments since it relies on GPS. Our solution behaves like a multi-dimensional distributed index, but its indexing feature is cross-layered with routing, meaning that no physical position nor any external routing protocol is necessary, routing information is given by the index itself.

In [6] and [14] data space partitions, whose splitting method derives from [12], follow the physical positions of nodes, instead of the distribution of data. The consequence is that the storage load per node is, in general, unbalanced, because

it depends on the physical network topology; this leads to an unbalanced number of routings among nodes, particularly with not random data as shown by the experiments, and consequently to a rapid network break-up caused by premature turning off of most loaded sensors. In W-Grid the storage load balancing has been achieved thanks to two key points: (i) the multi-dimensional data space partitions occur according to the actual data distribution and (ii) each partition has the same maximum bucket size. Another key difference is that data partitions in [6] and [14] are disjoint, while in W-Grid they are nested. The main difference between [6] and [14] is that the latter requires a fewer number of sensors with GPS.

3 W-Grid

From now on, in this paper we will use the term nodes and sensors interchangeably. The main idea is to map sensors on a binary tree so that the resulting coordinate space reflects the underlying connectivity among them. Basically we aim to set parent-child relationships to the sensors which can sense each other, in this way we are always able to route messages, in the worst cases simply following the paths indicated by the tree structure. Using virtual coordinates that do not try to approximate node's geographic position we eliminate any risk of dead-ends. Basically W-Grid can be viewed as a binary tree index cross-layering both routing and data management features in that, (1) by implicitly generating coordinates and relations among nodes allows efficient message routing and, at the same time, (2) the coordinates determine a data indexing space partition for the management of multi-dimensional data. Each node has one or more virtual coordinates on which the order relation is defined and through which the routing occurs, and at the same time each virtual coordinate represents a portion of the data indexing space for which a device is assigned the management responsibility. W-Grid virtual coordinates are generated on a one-dimensional space and the devices do not need to have knowledge of their physical location. Thus, differently from algorithms based on geographic routing (see section 2), W-Grid routing is not affected by dead-ends. Since in sensor networks the most important operations are data gathering and querying it is necessary to guarantee the best efficiency during these tasks.

3.1 Generation of Virtual Coordinates

When a device, let us say d turns on for the first time, it starts a wireless channel scan (beaconing) searching for any existing W-Grid network to join (namely any neighbor device that already holds W-Grid virtual coordinates). If none W-Grid network is discovered, d creates a brand new virtual space coordinate and elects itself as root by getting the virtual coordinate "$*$"[1]. On the contrary, if beaconing returns one or more devices which hold already a W-Grid coordinate, n will join the existing network by getting a virtual coordinate.

[1] It is conventional to label "$*$" the root node.

Coordinate Setup. Whenever a node needs a new W-Grid coordinate, an existing one must be split. A new coordinate is given by an already participating node d_g, and we say that its coordinate c is split by concatenating a 0 or a 1 to it. The result of a split to c will be $c' = c+1$ and $c'' = c+0$. Then, one of the new coordinates is assigned to the joining node, while the other one is kept by the giving node. No more splits can be performed on the original coordinate c since this would generate duplicates. In order to guarantee coordinates' univocity even in case of simultaneous requests, each asking node must be acknowledged by the giving one d_g. Thus, if two nodes ask for the same coordinate to split, only one request will succeed, while the other one will be canceled.

Coordinate Selection. At coordinate setup, if there are more neighbors which already participate the W-Grid network, the joining sensor must choose one of them from which to take a coordinate. The selection strategy we adopt is to choose the shortest coordinate[2] in terms of number of bits. If two or more strings have the same length the sensor randomly chooses one of them. Experiments have shown that this policy of coordinate selection reduces as much as possible the average coordinates length in the system. In Figure 1 there is a small example of a W-Grid network. In the tree structure, parent-child relationships can be set only by nodes that are capable of bi-directional direct communication.

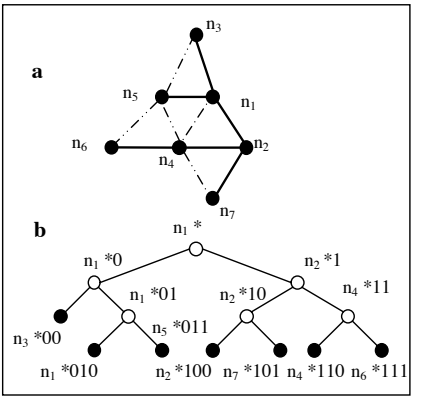

Fig. 1. Physical (a) and logical (b) network. Empty circles represent split coordinates, full black circles are coordinates that can still be split.

3.2 Formal Model: Network Properties

The sensor network is represented as a graph S:

$$S = (D, L)$$

[2] Among the ones that still can be split, see Coordinate Setup.

in which D is the set of participating devices and L is the set of physical connectivity between couples of devices:

$$L = \{(d_i, d_j) : two - way\ connection\ between\ d_i\ and\ d_j\}$$

Each device is assigned one or more (virtual) coordinate(s). We define C as the set of existing coordinates. Each coordinate c_i is represented as a string of bits starting with $\star$. According to the regular expression formalism coordinates are defined as follows:

$$C = \{c :\ c = \star(0 \mid 1)^*\}$$

E.g. $\star 01001$ is a valid W-Grid coordinate. Given a coordinate c_i and a bit b their concatenation will be indicated as $c_i b$. E.g. considering $c_i = \star 0100, b = 0$ then $c_i b = \star 01000$. Given a bit b its complementary $\overline{b}$ is defined. E.g $\overline{1} = 0$. Some functions are defined on C:

$$length(c) : C \rightarrow\ \mathbb{N} \tag{1}$$

Given a coordinate c, $length(c)$ returns the number of bits in c. ($\star$ excluded). E.g. $length(\star 01001) = 5$.

$$bit(c, k) : (C, \mathbb{N} - \{0\}) \rightarrow\ \{0, 1\} \tag{2}$$

Given a coordinate c and a positive integer $k \leq length(c)$, $bit(c, k)$ returns the k-th bit of c. Position 0 is out of the domain since it is occupied by $\star$.

$$pref(c, k) : (\mathbb{C}, \mathbb{N}) \rightarrow\ C \tag{3}$$

Given a coordinate c and a positive integer $k \leq length(c)$, $pref(c, k)$ returns the first k bits of c. E.g. $pref(\star 01001, 3) = \star 010$. We define the complementary (buddy) of a coordinate c as:

$$\overline{c} = pref(c, length(c) - 1)\overline{bit(c, length(c))} \tag{4}$$

E.g. $\overline{\star 01001)} = \star 01000$.

$$father(c) : (C - \{\star\}) \rightarrow\ C$$

$$father(c) = pref(c, length(c) - 1) \tag{5}$$

$$lChild(c), rChild(c) : (C) \rightarrow\ C$$

$$lChild(c) = c0 \tag{6}$$

$$rChild(c) = c1 \tag{7}$$

E.g. Given a coordinate $c_i = \star 011$, $father(\star 011) = \star 01$, $rChild(\star 011) = \star 0111$, $lChild(\star 011) = \star 0110$. A function M *maps* each coordinate c to the device holding it:

$$M : C \rightarrow D$$

A W-Grid network is represented as a graph:

$$W = (C, P)$$

P is the set of *parentships* between coordinates.

$$P = \{(c_i, c_j) : c_j = c_i(0 \mid 1)\}$$

E.g. $p_i = (\star 010, \star 0101)$. We define the complementary (buddy) of a parentship $p = (c_i, c_j)$ as:

$$\overline{p} = (c_i, \overline{c_j}) \tag{8}$$

E.g. $p = (\star 010, \star 0101), \overline{p} = (\star 010, \star 0100)$. A graph W is a valid W-Grid network if both the following properties are satisfied:

1. $\forall p = (c_i, c_j) \in P, (M(c_i) = M(c_j)) \vee ((M(c_i), M(c_j)) \in L)$
2. $\forall p = (c_i, c_j) \in P : M(c_i) \neq M(c_j) \Rightarrow \exists\ \overline{p} = (c_i, \overline{c_j}) \in P : M(c_i) = M(\overline{c}_j)$

3.3 Formal Model: Network Generation

W-Grid network is generated according to this few simple rules:

1. The first node that joins the networks (that initiate a coordinate space) gets the coordinate $\star$. A node that holds a W-Grid coordinate is marked as **active**. A function *last* is defined:

$$last(d) : D \rightarrow\ C$$

which returns the last coordinate received by d. If d is **not active** the function returns $\{\emptyset\}$. After the first node, let us say n_1, has joined the network, $last(n_1) = \star$.
2. $\forall\ l = (d_i, d_j) \in L : last(d_i) \neq \{\emptyset\}$ two parentships are generated:

- $p = (last(d_i), c')$: $M(c') = d_j$
- $\overline{p}$

Where $c' = lChild(last(d_i)) \mid rChild(last(d_i))$. Namely c' corresponds to the non-deterministic choice of one of the children of c. Nodes progressively get new coordinates from their physical neighbors in order to establish parentships with them. The number of coordinates at nodes may vary, in W-Grid that measure is always used as a parameter. The policies for coordinates may be: (1) a fixed number of coordinates per node (e.g. a given k) or (2) one coordinate per physical neighbor. Coordinates getting is also called split. The actors of the split procedure are an asking node and a giving node. A coordinate c_i is split by concatenating a bit to it and then, one of the new coordinates is assigned to the joining node, while the other one is kept by the giving node. Obviously, an

already split coordinate c_i can not be split anymore since this would generate duplicates. Besides, in order to guarantee coordinates' univocity even in case of simultaneous requests, each asking node must be acknowledged by the giving node. Thus, if two nodes ask for the same coordinate to split, only one request will succeed, while the other one will be temporarily rejected and postponed. Coordinate discovering is gradually performed by implicit overhearing of neighbor sensors transmissions.

3.4 Routing Algorithm

W-Grid maps nodes on an indexing binary tree T in order to build a totally ordered set over them. Each node of the tree is assigned a W-Grid virtual coordinate (c) which is represented by a binary string and has a value $v(c)$:

$$\forall\, c \in T, v(c) \in C$$

where C is a totally ordered set since:

$$\forall\, c_1, c_2 \in T : c_2 \in l(c_1) \rightarrow v(c_2) < v(c_1)$$

$$\forall\, c_1, c_2 \in T : c_2 \in r(c_1) \rightarrow v(c_2) > v(c_1)$$

where $r(c)$ and $l(c)$ represents the right sub-tree and the left sub-tree of a coordinate $c \in T$ respectively. And:

$$\forall\, c_1, c_2 \in T : F(c_1, c_2) = 0 \rightarrow v(c_1) < v(c_2)$$

$$\forall\, c_1, c_2 \in T : F(c_1, c_2) = 1 \rightarrow v(c_1) > v(c_2)$$

where $F(c_1, c_2)$ is a function that returns the bit of coordinate c_1 at position $i+1$ where i corresponds to the length of the common prefix between c_1 and c_2. For instance given two coordinates $c_1 = \mathbf{11}0100$ and $c_2 = \mathbf{11}10$, $F(c_1, c_2) = 0$[3] therefore $c_2 > c_1$. As we stated before, the coordinate creation algorithm of W-Grid generates an order among the nodes and its structure is represented by a binary tree. The main benefit of such organization is that messages can always be delivered to any destination coordinate, in the worst case by traveling across the network by following parent-child relationship. The routing of a message is based on the concept of distance among coordinates. The distance between two coordinates c_1 and c_2 is measured in logical hops and correspond to the sum of the number of bits of c_1 and c_2 which are not part of their common prefix. For instance:

$$d(\mathbf{*0}011, \mathbf{*0}11) = 5$$

Obviously it may happen that physical hops distance is less then the logical. Given a message and a target binary string c_t each node n_i forwards it to the neighbor that present the shortest distance to c_t. It is important to notice that each node needs neither global nor partial knowledge about network topology to route messages, its routing table is limited to information about its direct neighbors' coordinates. This means **scalability** with respect to network size.

[3] While $F(c_2, c_1) = 1$, therefore $F(c_1, c_2) = \overline{F(c_2, c_1)}$.

4 W-Grid Data Management

W-Grid organizes nodes (i.e. sensors/devices) in a tree structure and distributes data (tuple or records with any kind of information) among them by translating the values of the record attributes into binary strings, namely into virtual coordinates that are used to locate the matching node where to store the strings, that is the data. The translation of record values into binary strings occurs by means of a linearization function mapping multidimensional data to one dimension with a good locality preserving behavior. Several linearization functions, such as Z curve, Hilbert curve etc., have been successfully adopted in the past for multi-dimensional data structures (see [2] for a survey) and in particular we adopted a modified version of the one proposed in [12].

Since W-Grid c_i are binary strings, we can see from Figure 2 that they correspond to leaf nodes of a binary tree. Therefore a W-Grid network acts directly as a distributed database with a distributed index. This means that each coordinate represent a portion (i.e. region) of the global data space as depicted in Figure 2. The mechanism described in subsection 3.3 and in 3.1, which generates new coordinates, corresponds to a split method that creates also new regions. Basically, from the viewpoint of data management, this split method divides the region in two half of equal volumes along a space dimension. The dimension is chosen following a simple rule: if a region r has been achieved by splitting his father region along the i-th dimension, then r will be split on the successive dimension, namely *i-th+1 modulo number-of-space-dimensions*.

An additional concept related to region splits, which is specific of the data management feature, is that all region have a maximum bucket size b that fixes the maximum number of data managed by each region. When the number of any

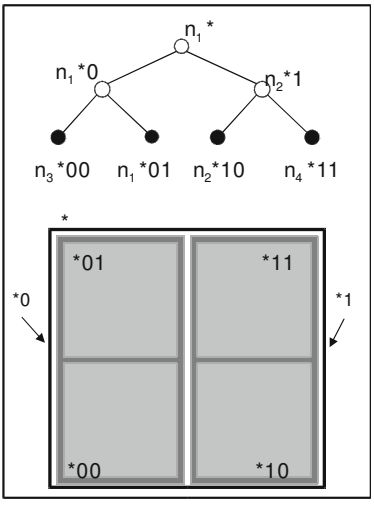

Fig. 2. Correspondence between coordinates and data space partitions

region data is equal to $b+1$ (region overflow) then the region undergoes a split following the same method just described, but with a slight difference: if after the split one of the resulting region is still in overflow, then the split process continues recursively and stops when no region is overflowing its bucket. The process converges quickly because the region is always divided in two half and moreover it is sufficient to separate only one (overflowing) region data from the other.

The region bucket size allows also a first indirect balancing of the storage load of regions at nodes, moreover each nodes may receive several coordinates/regions. Coordinates that have been split (the empty circles in Figures 1 and 2) cannot contain data.

Let us describe a brief example of an environment monitoring application in which sensors survey temperature (T) and pressure (P), to which we refer as d_1 and d_2. Each event is inserted in the distributed database implicitly generated by W-Grid, reporting for instance date and time of occurrence.

Without loss of generality we can define a domain for T and P let us say $Dom(d_1) = [-40, 60]$ and $Dom(d_2) = [700, 1100]$. We present an example of range query submitted to the network. *Return the events having a temperature ranging from 26 to 30 Celsius degrees and pressure ranging from 1013 to 1025mbar.* After calculating the correspondent binary string[4] for the four corners of the range query, namely:

$$\begin{array}{c} (26{,}1013)\ (26{,}1025)\ (30{,}1013)\ (30{,}1025) \\ c_1 = \textit{*11011}00\ c_2 = \textit{*11011}01 \\ c_3 = \textit{*11011}10\ c_4 = \textit{*11011}11 \end{array}$$

all we have to do is querying sensors whose coordinates have $*110110$ as prefix.

To do this we will route the range query toward $*110110$. Once the correspondent sensor has been reached it will be in charge to (1) solve part of the query if it is managing regions covered by the range query and (2) find out which of its child nodes (neighbor nodes) has coordinates that are covered by the range query. The query is then forwarded to each of these child node for further solving. We have fully implemented this algorithm and its performances are reported in Section 6.

5 Local Learning

This method introduces a learning algorithm locally at sensors, with no extra cost in terms of radio transmissions, whose goal is to learn information regarding direct sensor neighbors. This strategy improves routing performances by reducing the number of hops, namely the number of forwards, and consequently the

[4] For instance, by standardizing 26 and 1013 (c_1) to their domains we obtain $0, 66$ and $0, 783$ respectively. We multiply both of them by 2^4 in order to get a string of length 8. The binary conversion of the multiplications are **1010** and *1100* respectively. Then, by crossing bit by bit the two string we get the c where destination node location is stored *$\mathbf{1}\textit{1}\mathbf{0}\textit{1}\mathbf{1}\textit{0}\mathbf{0}\textit{0}$.

routing latency. The basic idea is to exploit the implicit overhearing that radio communications cause. In fact every time that a packet p_i (data or query) with destination c_i is forwarded by a sensor d_f (forwarder) to a sensor d_r (receiver), each sensor that is within the radio range (neighbors N) of d_f is aware that the packet is being forwarded. As a consequence each sensor in N can discover (i) which virtual coordinate is the destination of p_i, (ii) which sensor d_r has been chosen towards that destination and (iii) which is the distance of p_i at d_r. Here comes the local learning. If any sensor in N, let us say d_l finds out that a neighbors, let us say d_{nf} with coordinate c_{nf} would have taken p_i closer than d_r then d_l temporarily stores the pair (d_{nf}, c_i) so that when it performs the next beaconing it informs d_f that a better path has been discovered. In this way, the next time that d_f needs to forward a packet to a destination whose prefix is c_i, d_l will be preferred to d_r. Figure 3 shows an example of local learning. Packet p_i with destination $*011$ must be routed by node d_f. By forwarding p_i to d_l the distance from d_f to the destination is 3 while by forwarding p_i to d_f the distance is 5. Local learning allows n_f to know that d_l is a better choice for routing packets whose destination is $*011\%$[5].

A possible variation of the strategy is to choose between d_l and d_r according to a certain probability, so that possible changes in network topology and consequently new possible paths can be caught even if some learning has already occurred.

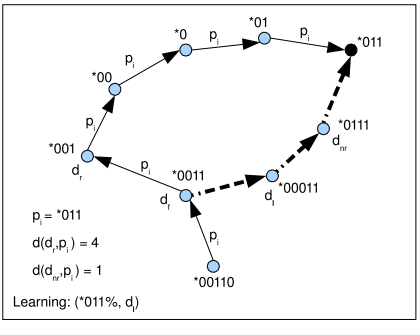

Fig. 3. Example of local learning at node n_f

6 Experimental Results

We have compared the performances of W-Grid algorithm with DIM [6] by implementing DIM in our Java Network Simulator.

We simulated four kinds of network deployment on an area of 800×800 meters in which 205 sensors where spread according to (1) uniform and (2) not uniform distribution and in both cases we generated two set of data based (a) on a random and (b) on a skewed distribution respectively. We varied nodes

[5] % means a binary string of arbitrary length.

densities by adjusting nodes transmission range (73, 101 and 122 meters) so that each sensor could have, on average, 4, 8 and 12 neighbors respectively. Sensors performed periodic beaconing so that coordinate creation was gradual, the simulation randomly chose one sensor to beacon first and elect itself as root of a virtual coordinate space. Then, as described in Section 3 we let sensors build the W-Grid network and a DIM network as well. Once both W-Grid and DIM network generation were completed we performed 2000 data insertion, with data generated into domains $D_1 = \{0, 800\}$ and $D_2 = \{0, 800\}$. After that we randomly generated 5000 range queries and injected them into the network to randomly chosen sensors. When creating a range query we followed these steps:

- Generation of a query central point (x, y) on D_1 and D_2
- Generation of the range by using Math.Gaussian Java function and multiplying the resulting value by a factor 70
- Applying the range to (x, y)

By fixing the factor to 70 we obtain that about 67% of the queries will have a range within 140 and 99% of them will have a range within 420. From simulations results we obtained that the 5000 range queries looked for 100000 data on average, meaning that each query covered an average of 20 data.

6.1 Network Traffic Comparison

When comparing DIM and W-Grid it is appropriate to make some considerations. DIM relies on GPSR when performing routing, this means that sensors need to be aware both of their physical location and the network perimeter. These constraints increase the cost of each sensor and limit the DIM usage possibility, for instance it cannot be used in indoor environments and in outdoor

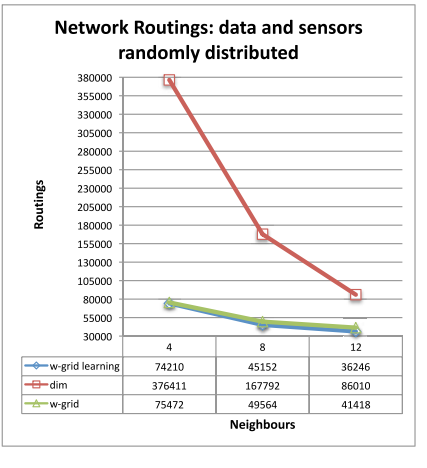

Fig. 4. Number of routings with sensors and data randomly distributed

areas where the density of sensors is beyond the GPS precision, or when weather conditions are bad.

W-Grid achieves significatively better routing performances than DIM in all of the four scenarios achieved by combining the two distributions of sensors with the two distributions of data. Moreover, the local learning improves the performance in all experiments achieving the best gain of 12% when not random data are distributed over randomly positioned sensors (see Figure 5). In all scenarios DIM reduces the wide gap to W-Grid as the network density increases. As depicted

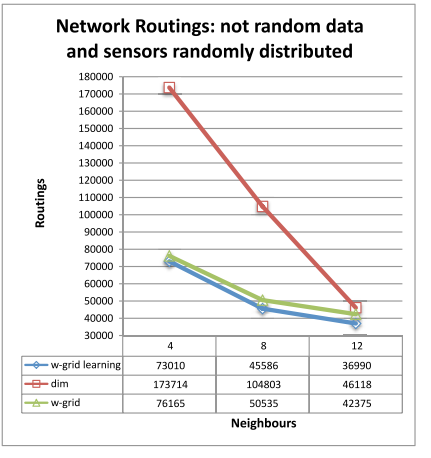

Fig. 5. Number of routings with sensors randomly distributed and data not randomly distributed

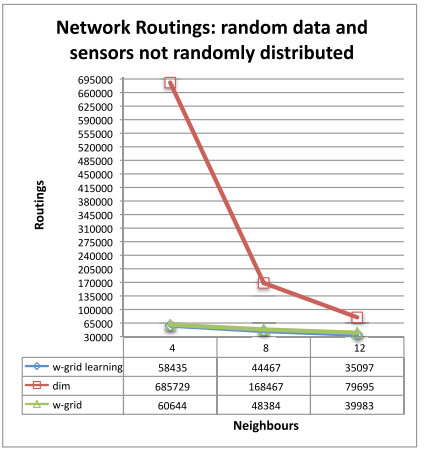

Fig. 6. Number of routings with sensors not randomly distributed and data randomly distributed

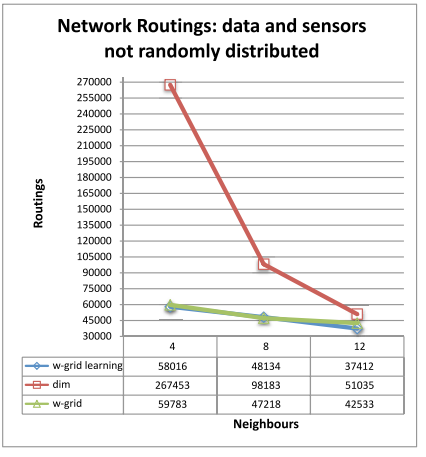

Fig. 7. Number of network routings with sensors and data not randomly distributed

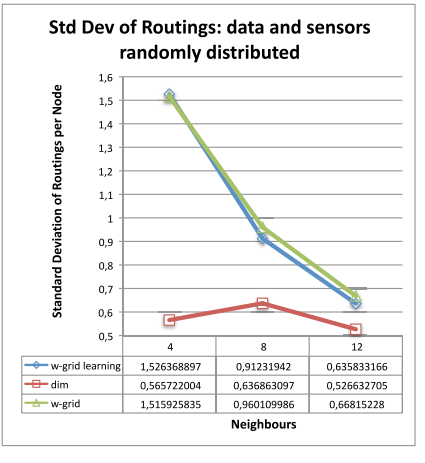

Fig. 8. Std Dev of routings over Avg routings per node, with sensors and data randomly distributed

in Figure 5 and in Figure 6, when the sensor density is 4 neighbors per sensor, DIM requires, respectively, between 3 and 10 times more routings (i.e. message forwards) than W-Grid in order to resolve the same sets of range queries over the same sensor deployments.

As far as the distribution of the routing workload per node is concerned, it is measured as the ratio between the standard deviation of the number of routings and the average of routings. When the standard deviation is greater than its corresponding average, the ratio is greater than 1 and of course it is smaller than 1 in the opposite case. It is necessary to adopt such a ratio to compare the W-Grid and DIM routing workload because the two approaches generate a

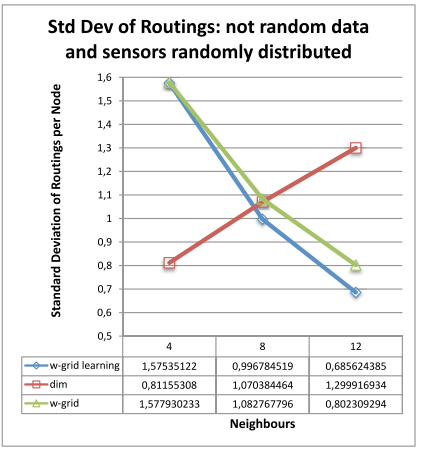

Fig. 9. Std Dev of routings over Avg routings per node, with sensors randomly distributed and data not randomly distributed

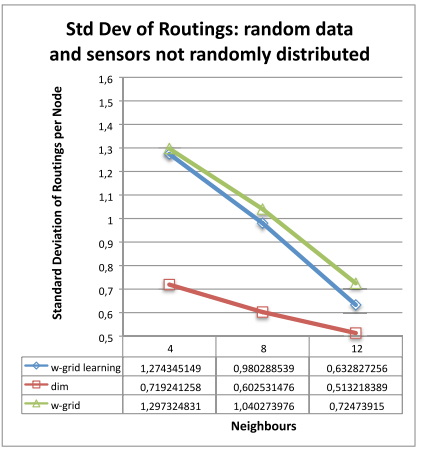

Fig. 10. Std Dev of routings over Avg routings per node, with sensors not randomly distributed and data randomly distributed

different number of routings for the same simulation configurations. As depicted in Figure 8 and 10 DIM behaves better than W-Grid when data are random. If data are not uniformly distributed, as it usually happens in real applications, W-Grid achieves a better workload balancing when the density is equal or greater than 8 neighbours per node (see Figure 9 and 11. Moreover the learning method always improves the routing workload balancing.

With regard to range queries efficacy we can observe in Figure 12 that a percentage of data between 2% and 3% are not cought by DIM range queries, while W-Grid does not miss any data. DIM losses are due to sensor placement

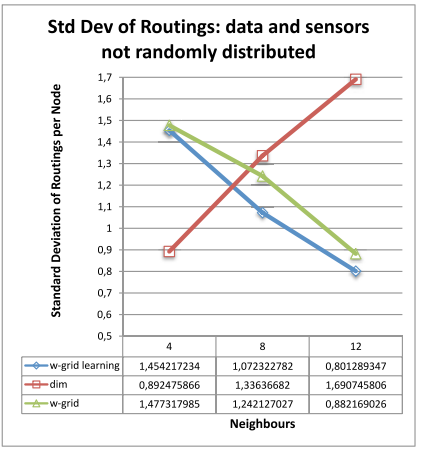

Fig. 11. Std Dev of routings over Avg routings per node, with sensors and data not randomly distributed

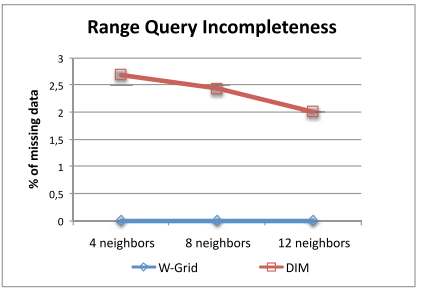

Fig. 12. Number of data not found by range queries

which may cause some regions not to be managed by any sensor and GPSR routing not being able to find the correct backup zone.

7 Conclusions

W-Grid is a cross-layering infrastructure able to self-organize wireless sensor networks for routing and multi-dimensional data management. Simulations have shown that W-Grid generates wireless networks, which significantly reduce the network traffic with respect to DIM networks in all the experimented scenarios. Moreover W-Grid produces a better balancing of the routing workload when data are not uniformly distributed and when the sensor density is equal or greater than 8 neighbours per sensor. Finally, the local learning method has further improved both the network traffic and the routing balancing of W-Grid in all experiments.

References

1. Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. In: Tan, K.-L., Franklin, M.J., Lui, J.C.-S. (eds.) MDM 2001. LNCS, vol. 1987, pp. 3–14. Springer, Heidelberg (2000)
2. Gaede, V., Günther, O.: Multidimensional access methods. ACM Computing Surveys 30(2), 170–231 (1998)
3. Greenstein, B., Estrin, D., Govindan, R., Ratnasamy, S., Shenker, S.: Difs: A distributed index for features in sensor networks. In: Proceedings of first IEEE WSNA, pp. 163–173. IEEE Computer Society, Los Alamitos (2003)
4. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. IEEE/ACM Trans. Netw. 11(1), 2–16 (2003)
5. Karp, B., Kung, H.: GPSR: greedy perimeter stateless routing for wireless networks. In: MobiCom 2000: 6th annual international conference on Mobile computing and networking, pp. 243–254. ACM Press, New York (2000)
6. Li, X., Kim, Y., Govindan, R., Hong, W.: Multi-dimensional range queries in sensor networks. In: SenSys 2003: Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 63–75. ACM Press, New York (2003)
7. Madden, S., Franklin, M., Hellerstein, J., Hong, W.: Tag: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. 36(SI), 131–146 (2002)
8. Monti, G., Moro, G.: Multidimensional Range Query and Load Balancing in Wireless Ad Hoc and Sensor Networks. In: Proceedings of the Eighth IEEE International Conference on Peer-to-Peer Computing (P2P 2008), pp. 205–214 (2008)
9. Monti, G., Moro, G.: Scalable multi-dimensional range queries and routing in data-centric sensor networks. In: Infoscale 2008: The Third International ICST Conference on Scalable Information Systems (2008)
10. Monti, G., Moro, G., Lodi, S.: W*-Grid a robust decentralized cross-layer infrastructure for routing and multi-dimensional data management in wireless ad-hoc sensor networks. In: P2P 2007: Seventh IEEE International Conference on Peer-To-Peer Computing, pp. 159–166 (2007)
11. Moro, G., Monti, G.: W-Grid: a self-organizing infrastructure for multi-dimensional querying and routing in wireless ad-hoc networks. In: P2P 2006: Sixth IEEE International Conference on Peer-To-Peer Computing, pp. 210–220 (2006)
12. Ouksel, M.A.: The interpolation based grid file. In: ACM SIGACT-SIGMOD 1985: Proceedings of Symposium on Principle of Database Systems, pp. 20–27. ACM Press, New York (1985)
13. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-centric storage in sensornets with ght, a geographic hash table. Mob. Netw. Appl. 8(4), 427–442 (2003)
14. Xiao, L., Ouksel, A.: Tolerance of localization imprecision in efficiently managing mobile sensor databases. In: MobiDE 2005: Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access, pp. 25–32. ACM Press, New York (2005)
15. Ye, F., Luo, H., Cheng, J., Lu, S., Zhang, L.: A two-tier data dissemination model for large-scale wireless sensor networks. In: MobiCom 2002: Proc. of the 8th annual international conference on Mobile computing and networking, pp. 148–159. ACM Press, New York (2002)