Scalable Multi-Dimensional Range Queries and Routing in Data-Centric Sensor Networks*

Gabriele Monti
DEIS - University of Bologna
Via Venezia, 52
Cesena I-47023
gabriele.monti4@unibo.it

Gianluca Moro
DEIS - University of Bologna
Via Venezia, 52
Cesena I-47023
gianluca.moro@unibo.it

ABSTRACT

Large scale wireless ad hoc networks of computers, sensors, PDAs etc. (i.e. nodes) are revolutionizing connectivity and leading to a paradigm shift from centralized systems to highly distributed and dynamic environments. An example of ad hoc networks are sensor networks, which are usually composed by small units able to sense and transmit to a sink elementary data which are successively processed by an external machine. Recent improvements in the memory and computational power of sensors, together with the reduction of energy consumptions, are rapidly changing the potential of such systems, moving the attention towards data-centric sensor networks. In these sensor networks, data or events are named by attributes that have scalar values, so one natural way to query events of interest is to use a multidimensional range query. W-Grid acts as a distributed index and allows multi-dimensional data management capability since nodes' virtual coordinates can act as a distributed database without needing neither special implementation or reorganization and supports multi-dimensional range queries. In this paper we introduce range query features in W-Grid and we show, by means of an extensive number of simulations, the performance and network costs in comparison with a well-know competitor solution in literature.

1. INTRODUCTION

Recent advances in information communication technology have led to the rapid development of small, powerful, multifunction devices with multi standard radio interfaces including Bluetooth, Wi-Fi and Wi-Max. For example, ad hoc networks are being designed where devices/nodes can directly communicate within a limited space both indoor, such as a building, and outdoor, such as a metropolitan area, without the need of a fixed pre-configured infrastructure and rigid data/communication protocols. These wireless ad hoc networks of computers, sensors, PDAs etc. (i.e. nodes) are revolutionizing connectivity and leading to a paradigm shift from centralized systems to highly distributed and dynamic environments. Compared to wired networks, wireless networks have unique characteristics. In wireless networks, nodes failure may cause frequent network topology changes, which 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 power restrictions and bandwidth limitations. Many different kinds of sensor applications have been studied in recent years. In these applications, a large volumes of data or events are continuously collected and stored by sensors for different kind of applications. From sensor networks literature Data-Centric Storage (DCS) scheme emerged to be the most efficient one for storing and processing data in a sensor network. In DCS, events are placed according to their "event" types, which refers to pre-defined attribute's values (temperature and pressure, for instance). Considering that data or events can be named by attributes or represented as relations in a distributed database [8] [1] [5], therefore a natural way to query for events of interest is using multi-dimensional range queries. Range queries can help users to efficiently drill-down their search or can be used by application software running within a sensor network for correlating events and triggering actions (e.g. a monitoring application). In traditional database systems, such range queries are supported using pre-computed indices. Indices trade-off some initial pre-computation cost to achieve a significantly more efficient querying capability. For sensor networks a centralized index for multi-dimensional range queries may not be feasible for energy-efficiency reasons (as well as the fact that the access bandwidth to this central index will be limited, particularly for queries emanating from within the network). Rather, we believe, there will be situations when it is more appropriate to build an in-network distributed data structure for efficiently answering multi-dimensional range queries. The W-Grid generates, in decentralized manner, virtual coordinates for each network device which reflect its local connectivity with other devices and uses this information to support message routing. These virtual coordinates also delineate the data space partition for which a device is assigned management responsibility, meaning that it is possible to distribute across the W-Grid network any kind of data. Basically W-Grid [11] [9] [10] is a binary tree index cross-layering both routing and data management features, in that (1) it al-

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

Infoscale 2008, June 4-6, 2008, Vico Equense, Napoli, Italy

lows efficient message routing and, at the same time, (2) the virtual 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 each virtual coordinate represents a portion of the data indexing space for which a device is assigned the management responsibility. As we will see from experimental results W-Grid range queries execution is scalable with respect to nodes density and distribution.

2. RELATED WORKS

A major limitation of the earlier DCS systems is that they either supported (support) only an exact match point query, such as GHT [14], or a simple range query concerning a maximum of two attributes, such as the time-indexing scheme [4] and the DIFS [3].

Advances in sensors hardware technology has brought the ability of detecting multiple types of parameters by each sensor. In order to monitor complex environments and phenomena it will be necessary to cope with more complex queries on multi-dimensional events. The DCS systems are not suited for managing and processing this kind of events. We need new techniques for coping with multi-dimensional events to answer multidimensional queries. This is because processing a query over multi-dimensional events is much more complex than processing a query on single-dimensional events. Some additional issues need to be addressed to cope with multi-dimensional events. First of all, events mapping must preserve the proximity of events since a bad mapping may highly increase data gathering. Then, it is important that the number of nodes to be queried does not increase with the number of sensors in the network.

So far, DIM (Distributed Index for Multi-dimensional data) [7] and [15] are the only solutions able to support multi-dimensional range queries in sensor networks. Both solutions adopt basically the same split mechanism of an historical centralized indexing method, the IBGF [13], and embed such an index in a sensor network. The data space partitions in both solutions cannot be nested. In DIM, each sensor is an index node. Thus, the number of index nodes in the sensor network depends on the size of the network. The sensor network is partitioned in zones which are assigned a unique zone code. The mapping between the value range and the zone code is accomplished by a locality preserving geographic hash. When answering a range query, DIM finds all zones whose value ranges overlap with the range in the query and sends the query to index nodes of those zones. The main drawbacks in DIM are that since it relies to GPSR[6] for routing queries it is subject to all the problems of geographic routing. More precisely, 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 experiments have shown that DIM performs really bad in case of skewed networks. The same issues hold for the solution in [15] which differs from DIM simply because they need a minor number of sensors equipped with GPS, anyway both solutions require that each sensor to know the physical coordinates of the network perimeter. Our solution does not require GPS and moreover the splitting mechanism we have ideated is different from both of them and from IBGF, in fact our data space partitions can be also nested in a manner closer to G-Grid [12], which, however, is a diverse multi-dimensional distributed indexing designed for overlay wired peer-to-peer systems.

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.

Coordinate Setup. Whenever a node needs a new W-Grid coordinate, an existing one must be split. The term "split" may seem misleading at the moment, but its meaning will become straightforward clear in Section 4. 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 $c' = c1$ and $c'' = c0$. 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

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

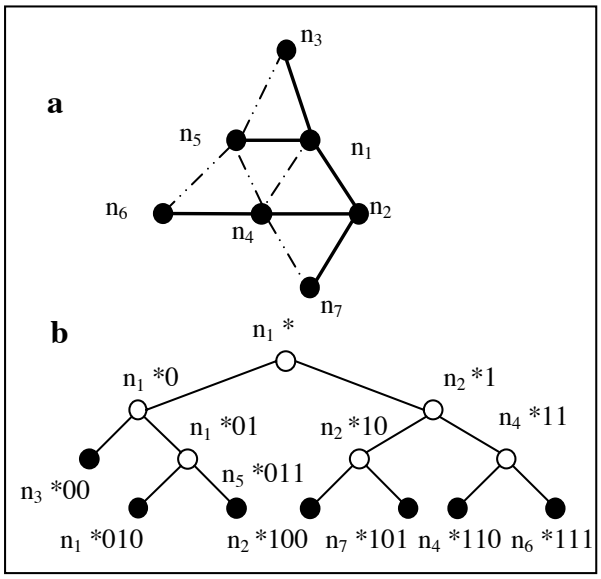

Figure 1: Physical (a) and logical (b) network. Empty circles represent split coordinates, full black circles are coordinates that can still be split.

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.

3.2 Formal Model: Network Properties

The sensor network is represented as a graph S:

$$S = (D, L)$$

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_ib. E.g. considering $c_i = \star0100, b = 0$ then $c_ib = \star01000$. 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. $length(\star01001) = 5$.

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

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. $pref(\star01001, 3) = \star010$. 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. $\overline{\star01001)} = \star01000$.

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

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

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

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

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

E.g. Given a coordinate $c_i = \star011$, $father(\star011) = \star01$, $rChild(\star011) = \star0111$, $lChild(\star011) = \star0110$. 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. $p_i = (\star010, \star0101)$. 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. $p = (\star010, \star0101)$, $\overline{p} = (\star010, \star0100)$. 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

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

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]

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

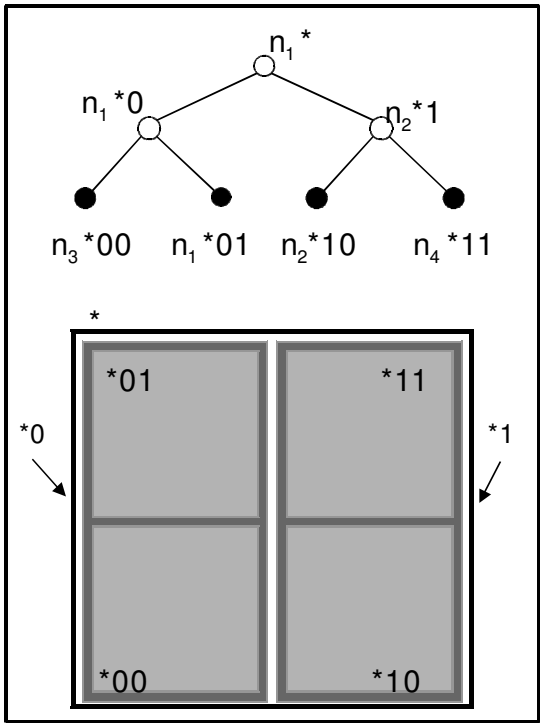

Figure 2: Correspondence between coordinates and data space partitions

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.

4. RANGE QUERIES IN W-GRID

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 [13].

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 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 basic indirect balancing of the storage load of regions at nodes, moreover we remind that each nodes may receive several coordinates/regions. Obviously 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:

(26,1013) (26,1025) (30,1013) (30,1025)

$c_1 =$ **11011000* $c_2 =$ **11011001*
$c_3 =$ **11011010* $c_4 =$ **11011011*

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 5.

5. EXPERIMENTAL RESULTS

In order to evaluate the performances of W-Grid algorithm compared with DIM, we extended our Java Network Simulator (netsim) by adding DIM algorithm and ran a set of simulations. 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 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.

5.1 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

[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 ****1***1***0***1***1***0***0***0*.

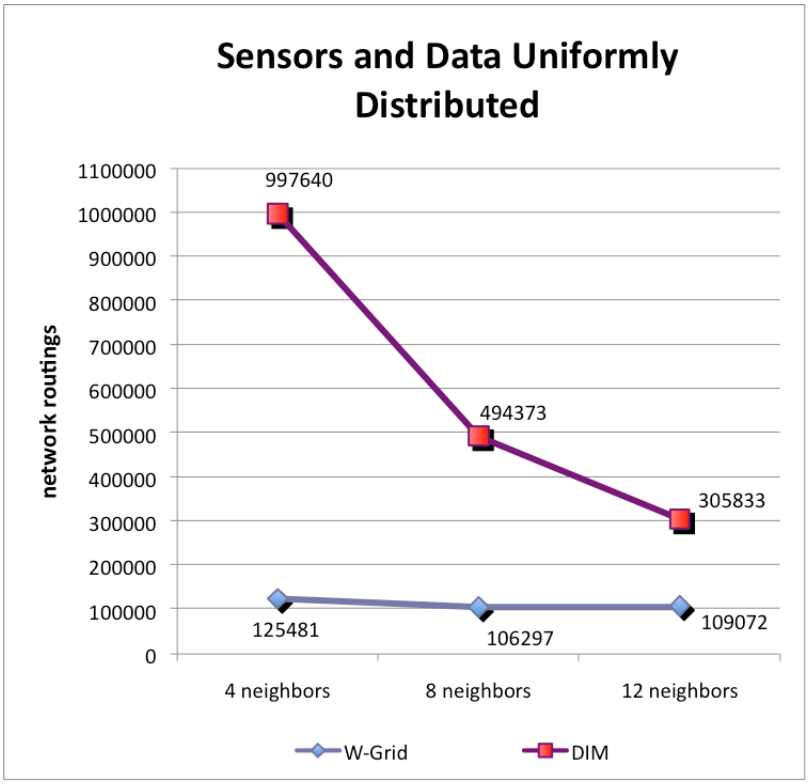

Figure 3: Number of network routings with sensors and data uniformly distributed

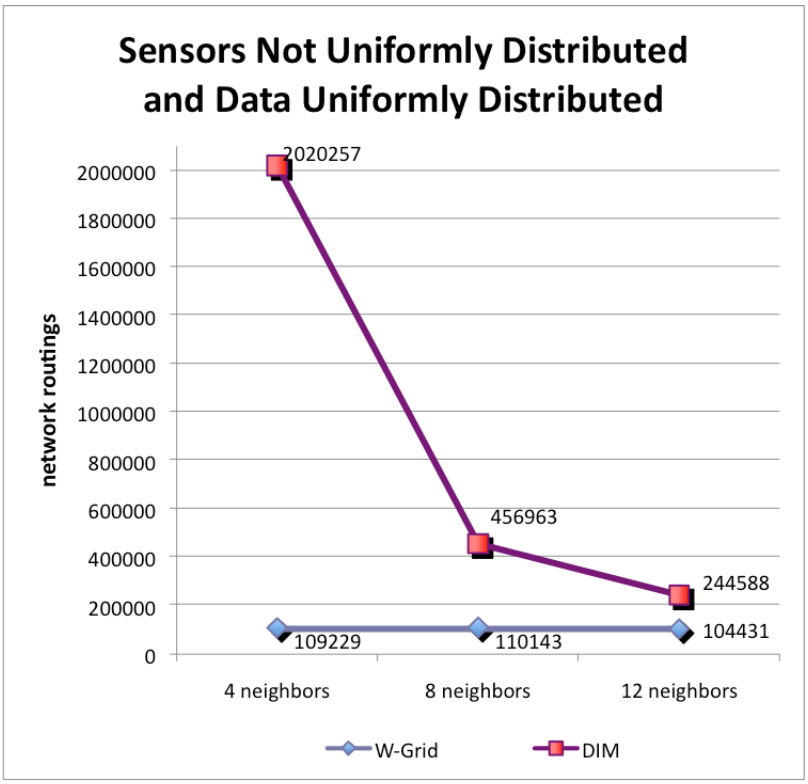

Figure 4: Number of network routings with sensors not uniformly distributed and data uniformly distributed

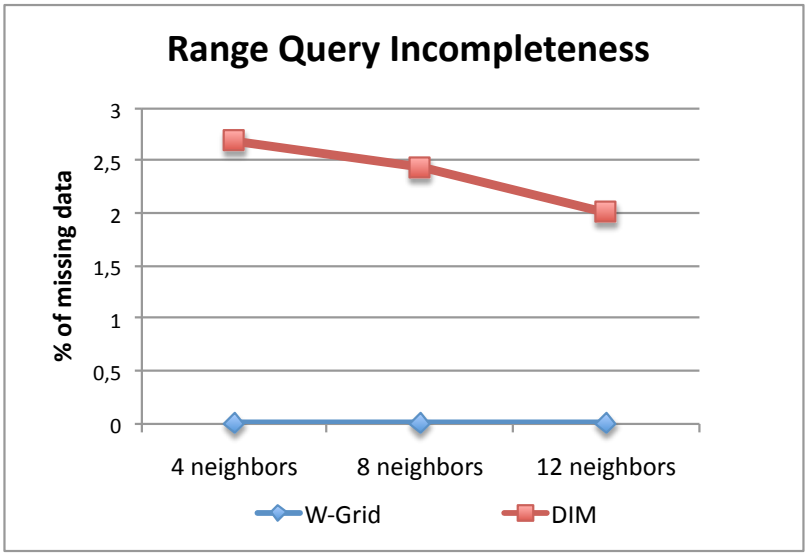

Figure 5: Number of data not found by range queries

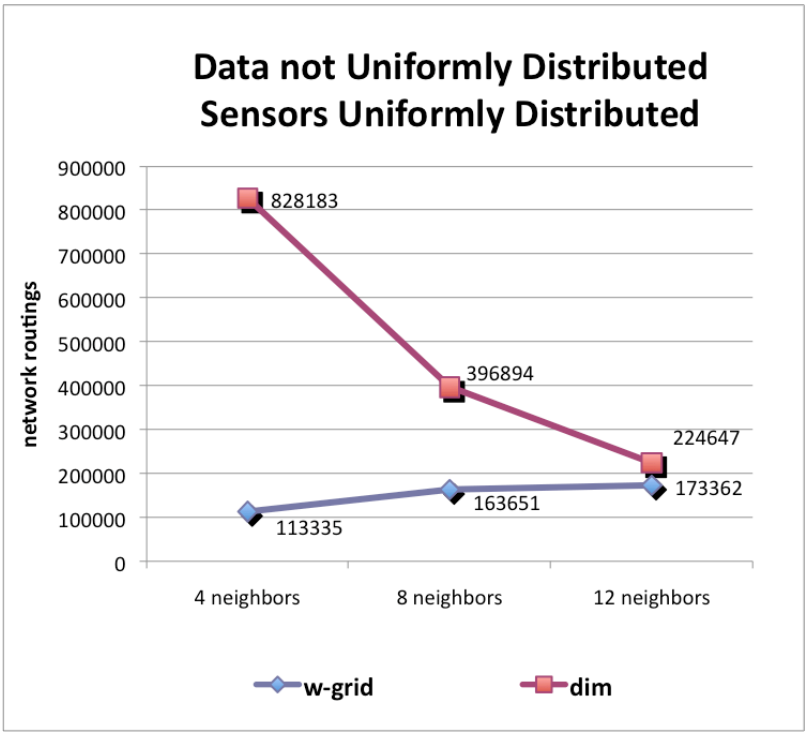

Figure 6: Number of network routings with sensors uniformly distributed and data not uniformly distributed

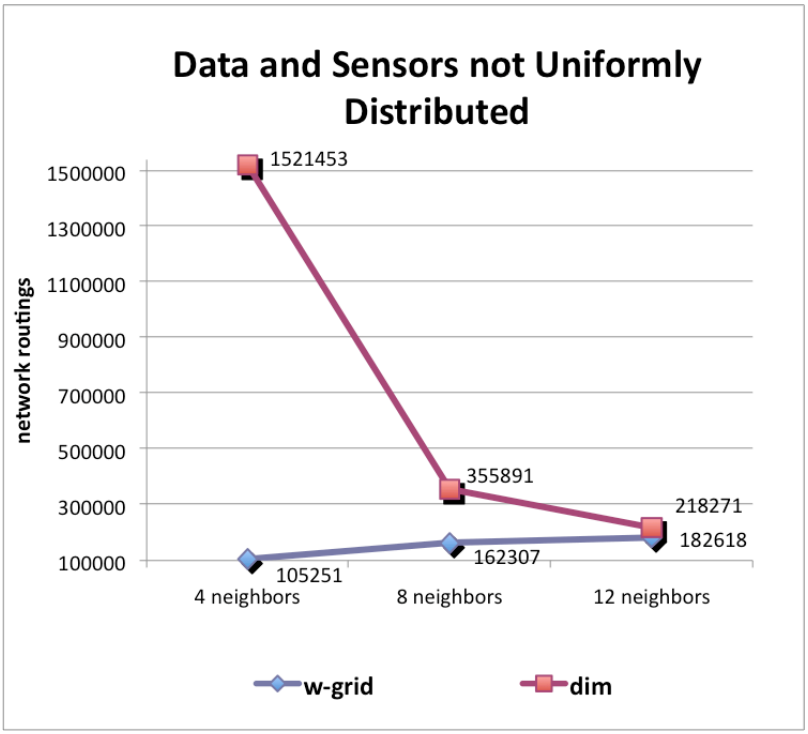

Figure 7: Number of network routings with sensors and data not uniformly distributed

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

W-Grid achieves significative better performance than DIM in all of the four scenarios combining the two distributions of sensors with the two distributions of data. In all scenarios DIM reduces the wide gap with respect to W-Grid as the network density increases. As depicted in Figure 3 and in Figure 4, when the sensor density is low, DIM requires, respectively, between 8 and 20 times more routings (i.e. messages) than W-Grid in order to resolve the same sets of range queries over the same sensor deployments. Both Figure 4 and Figure 7 show that DIM achieves the worst results in low density networks where sensors are not uniformly distributed. DIM achieves performance close to W-Grid only when the density is high, namely 12 neighbours on average per sensor, and data are not randomly/uniformly distributed, as illustrated in Figure 6 and Figure 7 respectively.

In summary W-Grid is really well suited for sensor network applications where sensor displacement is not defined a priori but random and/or in cases of low density. With regard to range queries efficacy we can observe in Figure 5 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 which may cause some regions not to be managed by any sensor and GPSR routing not being able to find the correct backup zone.

6. CONCLUSIONS

In this paper we presented range queries in W-Grid, a cross-layering algorithm for routing and data management, which acts as a distributed index and allows multi-dimensional data management and multi-dimensional range queries. W-Grid generates virtual coordinates at sensors that reflect the local connectivity among them and uses this information to support message routing. Besides, the virtual coordinates represent the data space partition for which a sensor is assigned management responsibility, meaning that it is possible to distribute across the W-Grid network any kind of data. After presenting W-Grid main characteristics and features we introduced how range queries can be performed in W-Grid and in Section 5 we showed, by means of an extensive number of simulations, that the W-Grid performances and network costs are much more better than DIM, a well-know competitor solution in literature.

7. REFERENCES

[1] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *MDM '01: Proceedings of the Second International Conference on Mobile Data Management*, pages 3–14, London, UK, 2001. Springer-Verlag.

[2] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[3] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. Difs: A distributed index for features in sensor networks. In *Proceedings of first IEEE WSNA*, pages 163–173. IEEE Computer Society, 2003.

[4] G. He, R. Zheng, I. Gupta, and L. Sha. A framework for time indexing in sensor networks. *ACM Trans.*

Sen. Netw., 1(1):101–133, 2005.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[6] B. Karp and H. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM Press, 2000.

[7] X. Li, Y. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 63–75, New York, NY, USA, 2003. ACM Press.

[8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[9] G. Monti, G. Moro, and S. Lodi. 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*, pages 159–166, 2007.

[10] G. Moro and G. Monti. 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*, pages 210 – 220, 2006.

[11] G. Moro, G. Monti, and A. Ouksel. Routing and localization services in self-organizing wireless ad-hoc and sensor networks using virtual coordinates. In *ICPS'06: IEEE International Conference on Pervasive Services 2006*, pages 243 – 246, 2006.

[12] G. Moro and A. Ouksel. G-Grid: A class of scalable and self-organizing data structures for multi-dimensional querying and content routing in P2P networks. In *Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing, Melbourne, Australia, July 2003*, volume 2872, pages 123–137. Springer, 2003.

[13] M. A. Ouksel. The interpolation based grid file. In *ACM SIGACT-SIGMOD '85: Proceedings of Symposium on Principle of Database Systems*, pages 20–27, New York, NY, USA, 1985. ACM Press.

[14] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mob. Netw. Appl.*, 8(4):427–442, 2003.

[15] L. Xiao and A. Ouksel. Tolerance of localization imprecision in efficiently managing mobile sensor databases. In *MobiDE '05: Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access*, pages 25–32, New York, NY, USA, 2005. ACM Press.