W-Grid: a Cross-Layer Infrastructure for Multi-Dimensional Indexing, Querying and Routing in Wireless Ad-Hoc and Sensor Networks

Gianluca Moro and Gabriele Monti

***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. A plethora of routing algorithms have been proposed for the network path discovery ranging from broadcasting/flooding-based approaches to those using global positioning systems (GPS). In this paper we propose a novel decentralized infrastructure that self-organizes wireless devices in an ad-hoc network, where each node has one or more virtual coordinates through which both message routing and data management occur without reliance on either flooding/broadcasting operations or GPS. The resulting ad-hoc network does not suffer from the dead-end problem, which happens in geographic-based routing when a node is unable to locate a neighbor closer to the destination than itself. The multi-dimensional data management capability will be described showing, as an example, how the location service reduces to a simple query, like for any other data type. Extensive performance analysis and experiments have been conducted and the results compared to GPSR, which is considered the most efficient routing solution not using broadcast operations. Our approach shows significant performance gains.**

I. Introduction

Recent advances in information communication technology have led to the rapid development of small, powerful, multi-function 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. Connectivity in this environment is supported by multi-hop transmission, where intermediate nodes act as signal repeaters according to a given routing strategy. The goal is to enable self-organizing ad-hoc networks, composed of wireless devices including sensors, which are virtually free from configuration and administration costs, and to support location and time sensitive applications in variety of domains. These include vehicle traffic, parking, public transport, commercial, security and emergency, and social networks.

A wide number of routing algorithms for ad-hoc networks have been proposed, ranging from those that adopt message broadcast/flooding to those using global position systems (GPS) to discover the routing path towards the destination. Broadcast algorithms, while simple to implement, are not scalable due to the enormous overhead caused by congestion in large networks. On the other hand, solutions based on GPS, which rely on exact geographic position for each node, does not work in indoor environments and does not function correctly in extremely dense networks or in adverse climatic conditions. Technical and economic feasibility constraints also prevent from attaching a GPS receiver to each node in very large networks (i.e. made of thousand of devices). For these reasons our solution does not rely on GPS or any other positioning system. The routing problem has also been addressed in cases of both total absence and partial availability of geographic location information by generating virtual coordinates to approximate real ones. Our solution may be classified within this set of approaches in that it also uses virtual coordinates, but it is distinctive in that it does not aim to approximate real coordinates.

Basically W-Grid [9] is a binary tree index cross-layering both routing and data management features, in that (1) by implicitly generating a total order relation among nodes it allows efficient message routing and, at the same time, (2) the order relation determines 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. Differently from algorithms based on geographic routing (see section II), W-Grid routing is not affected by dead-ends. To proof the routing and multi-dimensional data management features we will give a short description of a location service in which finding the location of a specific device reduces to a query over a distributed database.

Moreover W-Grid can also simply act as the routing network layer upon which existing indexing structures can be applied. For instance we think about the ones that were developed in the past for centralized environments (e.g. [4] and [13], see [3] for an extensive survey) and which have been extended in the last years to work in distributed environments, especially in wired overlay peer-to-peer networks [18] [20] [19] on top of TCP/IP layer of well-organized physical networks.

In this work we consider W-Grid to be used in wireless ad-hoc and sensor networks where, though nodes are not inherently mobile, each device can also disconnect from the network (e.g. failures). The paper is organized as follows. Section II discusses related works. In Section III, we describe the rules for generating the virtual coordinates and the routing algorithm, while data management issues are

Gianluca Moro and Gabriele Monti are with the Dept. of Electronic, Computer Science and Systems, University of Bologna
Viale Venezia, 52 47023 Cesena, Italy
email: gmoro,gmonti@deis.unibo.it

addressed in Section IV. Section V illustrates experimental results compared with GPSR algorithm [6], which is the most efficient solution not using broadcast operations. Section VI concludes the paper with open issues and perspective works.

II. Related Works

Routing protocols for ad-hoc networks are typically subdivided into three main categories: Table-driven (also known as proactive), On-Demand (or Reactive) and geographic routing protocols.

Table-driven routing protocols [15] [2] [11] recall the Internet distance-vector and link-state protocols. Nodes maintain tables that store routing information and any change in network topology triggers propagating updates in order to maintain a consistent view. This can cause heavy overhead affecting bandwidth utilization, throughput and power usage. The advantage of these protocols is that routes to any destination are always available without the latency of a route discovery, but on the other side, they do not perform properly when the number of participating nodes is high. The main differences in protocols belonging to this category are on the number of tables that nodes store and how they are updated.

On-demand routing protocols [16] [5] [14] are characterized by a path discovery mechanism that is initiated when a source needs to communicate with a destination that it does not know how to reach. Usually route discovery requires a form of query flood and for this reason on-demand routing incurs in a delay whenever a new path is needed. The differences between the several on-demand protocols are in the implementation of the path discovery mechanism.

A completely different approach is used by geographic routing protocols such as [6] [7]. The idea in geographical routing is to use a node's location as its address, and to forward packets with the goal of reducing as much as possible the physical distance to the destination. Geographic routing achieves good scalability since each node only needs to be aware of neighbors' position and because it does not rely on flooding to explore network topology. However it suffers of dead end problems, especially under low density environment or scenarios with obstacles or holes. Problems are caused by the inherent greedy nature of the algorithm that can lead to situations in which a packet gets stuck at a local optimal node 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 well-known 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. Another limitation of geographic routing is that it needs nodes to know their physical position. Usually authors assume that they embed GPS but it must be said that GPS receivers are expensive and energy inefficient compared to the devices that could participate in ad-hoc networks. Besides, GPS reception might be easily obstructed by climatic conditions or obstacles and doesn't work indoor.

Recently, virtual coordinates were proposed to exploit the advantages of geographic routing in absence of location information [17], [10], [1]. The motivation is that in many applications it is not necessary to know the exact coordinates but is often sufficient to have virtual coordinates that approximate real ones. Unfortunately virtual coordinate systems suffer the same dead end problem of standard geographic routing. W-Grid employs virtual coordinates like these last algorithms but it is based on a different approach which does not approximate real coordinates and eliminates the risk of dead-ends.

III. W-Grid: Indexing and Routing

We consider the case of nodes equipped with a wireless device. Each one is, at the same time, client of the network (e.g. sending messages), and responsible for managing others nodes communications (e.g. routing messages). For this reason from now on we will refer to them as nodes or peers indistinctly.

The main idea in W-Grid is to map 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 $c_2 = \mathbf{11}10$, $F(c_1, c_2) = 0$[1] therefore $c_1 < c_2$.

A. Virtual Coordinate Selection and Generation

When a node, let us say n 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, n creates a brand new space of virtual coordinates electing itself as root by getting the coordinate "$*$"[2]. On the contrary, if beaconing returns one or more devices which already hold W-Grid

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

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

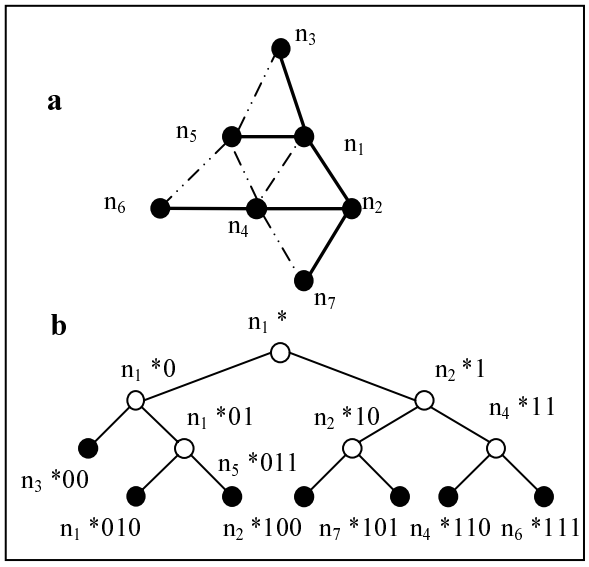

Fig. 1. Physical (a) and logical (b) network. Empty circles represent split coordinates, full black circles are coordinates that can still be split.

coordinates, n will join the existing network by getting a coordinate as well.

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 IV. A new coordinate is given by an already participating node n_g through an existing coordinate split. We say that a 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 n_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 node must choose one of them from which to take a coordinate. The selection strategy we adopt is to choose the shortest coordinate[3] in terms of number of bits. If two or more strings have the same length the node 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. This property is called *integrity* of coordinates and it is crucial for the network efficiency:

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

Definition 1: Let c be a coordinate at node n that has been split into c' and c''. Then we say that c is *integral* if either c' or c'' is held by a node $n' \in NEIGH(n)$, where $NEIGH(n)$ is the set of its neighbors.

If each coordinate satisfies this constraint, it will be possible to route any message, at least by following the paths indicated by the tree structure, without dead-ends.

B. Effects of Assigning Multiple Coordinates to Peers

When the number of dimensions of a space is reduced, some points of the space lose proximity. Since W-Grid virtual coordinates space is one-dimensional, while nodes are spread on a two-dimensional space (for simplicity we consider nodes to be at the same height), it means that two nodes physically close in the real space may be far away in the virtual space (e.g. they have very different virtual coordinates). As routing is performed through virtual coordinates surely it will lose efficiency whenever these situations occur. We came to the conclusion that it is possible to widely reduce inefficiencies be assigning more different coordinates to each node. In fact, having more than one coordinate means that a node is placed in different positions of the tree structure and this reduces the probability that two nodes physically close are very distant according to the order relation.

In Figure 2 each node is assigned a number of virtual coordinates equal to the number of its neighbors. Simulations returned that this coordinates generation policy ensures the best results in terms of combination between network efficiency and quantity of information stored at nodes. In fact there is a trade-off between these two measures since a higher number of coordinates per node translates into best routing performances but also implies larger routing tables and needs more storage capability at nodes.

In order to improve readability of the figure, for each node are shown only the coordinates that have not been split. The only exceptions are the coordinates interested by routing from node n_{17} to node n_{13}. This in useful to understand that split coordinates are stored at nodes and are used for routing. For instance node n_1, the root of the coordinate space, holds also coordinates *, *0 and *00; namely through multiple splits of root coordinate * we obtained *001.

C. Routing Algorithm

As we stated in the previous subsection, 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:

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

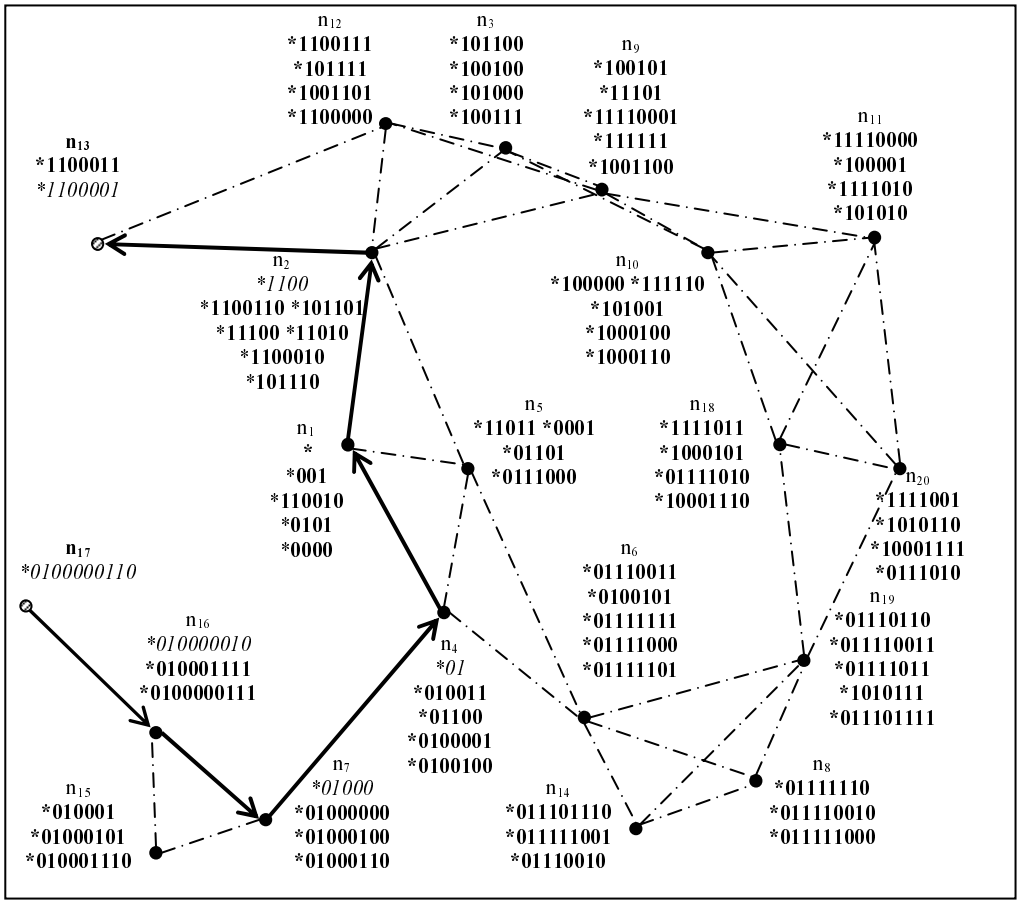

Fig. 2. A (small) example of a network with W-Grid coordinates and routing of a message with W-Grid (from node n_{17} to node n_{13}).

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 presents the shortest distance to c_t. It is important to notice that nodes need neither global nor partial knowledge about network topology to route messages, routing tables content is limited to information about direct neighbors' coordinates. This means **scalability** with respect to network size.

W-Grid metric has a very interesting feature. Given a virtual coordinate c and a distance d, there are several $c_i \in C$ which are distant d from c. For instance, given $*0011$ and distance 3:

$$d(*0011, *0) = 3$$
$$d(*0011, *000) = 3$$
$$d(*0011, *00100) = 3$$
$$\text{etc.}$$

In general given a coordinate c of length l, the number of coordinates whose distance from c is d is given by:

$$\sum_{\alpha=\max(1,l-d)}^{\max(1,l-1)} 2^{\Delta-1} \quad where\ \Delta = d - (l - \alpha) \qquad (1)$$

From (1) we can say that for each coordinate and distance there exist a set of coordinates at that distance that we call $c(d)$ (*distance set*). Thus, at each hop during the routing, a node n distant d from the destination has at least one neighbor that improves by one the distance (in logical hops) from the destination[4]. However, it is also possible that other neighbors of n belong to $c(d-1)$. This means a certain robustness to nodes failures and also the possibility of adopting specific and changeable policies for routing (for instance by forwarding to the node with most battery power left, in case of more nodes with the same distance from the target).

D. Nodes Failure

In ad-hoc networks nodes usually have scarce resource and they especially suffer of power constraints. This can lead to nodes failures that could affect routing efficiency. In W-Grid some robustness is guaranteed by multiple coordinates at each peer and by the adopted routing metric, that allow routing along several different paths. If a broken path is discovered the packet can change path (e.g. next hop), according to a different coordinate. However it may happen that a path breaks due to a node failure and no alternative way can be chosen.

[4] This is a consequence of coordinates integrity

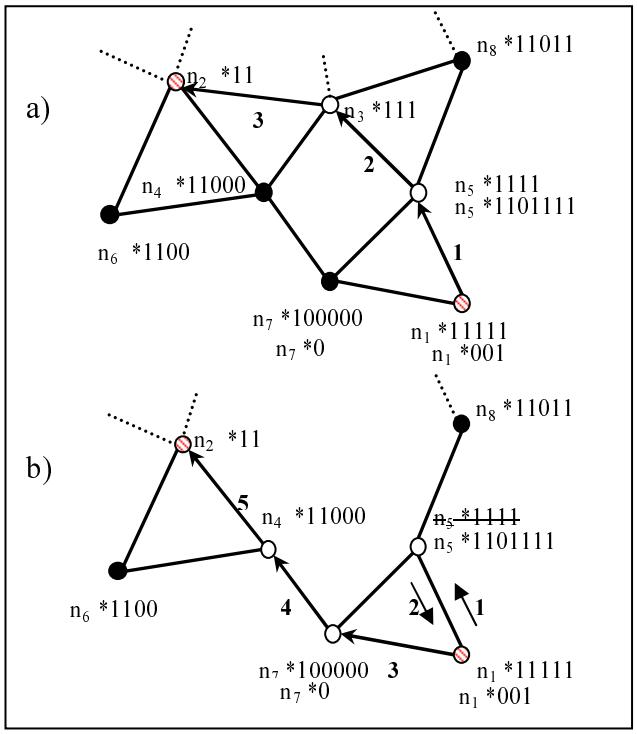

Fig. 3. Effects of node failure (n_3) during routing of a packet from node n_1 to n_2

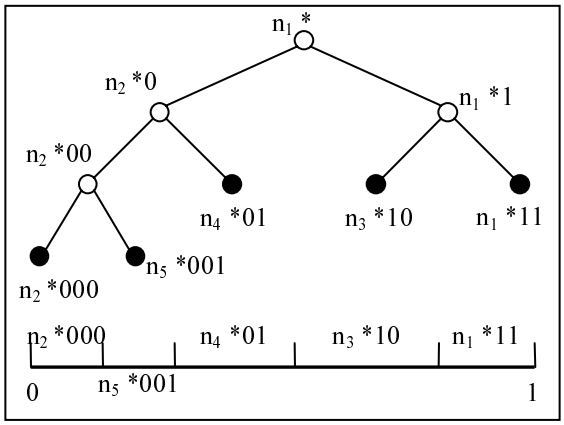

Fig. 4. Correspondence between W-GRID virtual coordinates and data space partitions.

In Figure 3 we present the case of a packet that must be routed to coordinate $*11$. During the routing a dead-end occurs, node n_5 cannot find any neighbor that improves its distance from the destination. This means that a link has broken since W-Grid total order relation guarantees the delivery in any case. When this happens the node deletes the coordinate that caused the dead-end and performs a "local broadcast" searching for the parent of the missing coordinate ($*11$ in our example). We use the term "local broadcast" since it is very likely that the searched coordinate will be close to the broadcasting node since it is a close relative of it. This means that the broadcast packet time-to-live will be small and its effects on network traffic will be limited. Once the coordinate has been found, the holding node fixes the relationship with the affected node by giving it a new coordinate, in our case through n_4 and n_7. It is important to specify that every recovery operation is lazy and triggered only on routing failures, in order to avoid any network efficiency loss.

IV. Data Management in W-Grid

W-Grid organizes peers in a tree structure and distributes data (tuple/records with any kind of information) among them by hashing the values of the record attributes into binary strings and storing them at peers whose W-Grid coordinates match the strings. Coordinates are binary strings and we can see from Figure 4 that they correspond to leaf nodes of a binary tree, therefore a W-Grid network acts directly as a distributed database. Obviously coordinates that have been split (the empty circles) cannot contain any data. One of the most important features that a distributed database must satisfy is a balanced storage load among the different nodes, even in case of not uniform distributions of data. In fact, if some nodes manage higher quantity of information they will likely receive a higher number of queries than the others causing bottlenecks and loss of efficiency for the entire network.

Due to the coordinates *integrity* constraint, related coordinates must belong to nodes that can directly contact each other. This means that each coordinate can split only a limited number of times (depending on the number of neighbors). Assuming uniform density of devices it is easy to understand that nodes managing shorter coordinates (likely the first nodes joining the network) will split about the same times of others nodes. However, since their initial regions were bigger, even after splits they will remain bigger than the ones of other nodes. It is easy to infer that this translates into a very unbalanced storage load situation.

In order to improve the data distribution balance we implemented the Storage Load Balancing (SLOB) Algorithm that will be described in Subsection A. Then in Subsection B we will show its effects on a real problem, namely the definition of a location service that provides information about the position, yet in terms of W-Grid virtual coordinates, of any participant. Basically, the location service is a usual exact match query on distributed data where there is a correspondence between data and nodes location.

A. Storage Load Balancing in W-Grid

To address the load balancing problem, existing in most of distributed data structures, we incorporate the concept of bucket size b namely the maximum number of data that a region (i.e. a coordinate) can manage. The value for b can be the same for each peer or, in environments where devices have different characteristics, it can be proportional for instance to the storage and/or communication bandwidth capabilities.

Whenever a node receives a new data it checks wether the space represented by the coordinate that must store the data is full or not. In case it is full the coordinate is split,

but, differently from what it happens when a new node joins the network, in this case both the resulting subspaces are stored at the peer.

The bucket size guarantees that each coordinate contains at most the same quantity of information. However, this trick does not balance the storage load on its own. In fact, peers holding spaces with a higher number of data will split more frequently than the others. The result will be that those peers will manage more coordinates if we do not find a way for them to give away the ones in excess, which is exactly the goal of Storage Load Balancing Algorithm (SLOB). On periodic beaconing each peer evaluates

Algorithm 1 Storage LOad Balancing Algorithm

$MyLoad \Leftarrow$ storage load at peer
scan neighbors and return $avgNeighLoad$, $neighLoadRMSE$ and $mostLoadedNeighbor$
if $(avgNeighLoad - MyLoad) > avgThreshold\, OR\, (avgNeighLoad > Load\ \&$ $RMSE > RMSEThreshold)$ **then**
  get one c from $mostLoadedNeighbor$
end if

the average storage load and the correspondent Root Mean Square Error ($avgNeighLoad$ and $neighLoadRMSE$ in algorithm 1) of its neighbors. The storage load of a node is meant as the number of coordinates held excluding split coordinates (not considered since there can be no data in them).

The purpose of this evaluation is discovering local unbalanced situations and moving a small step towards better balancing. In practice, a peer p_i compares its own load with the average, if the load is lower and the difference between the two measures is higher than a certain threshold (avgThreshold in algorithm 1) p_i takes one coordinate from the neighbor that has the highest storage load. A coordinate is taken anyway if the load is the same as the average but the RMSE is higher than a given threshold (RMSEThreshold in algorithm 1). The algorithm is as much simple as it is powerful since adding a local rule is able to create a global behavior that makes converge the network storage load toward a balanced situation.

B. Location service

Supposing that each peer n_i that composes the network is univocally identified by a public ID_i (such as the e-mail address, the MAC Address or any other unique ID) we can think about inserting in the distributed database, implicitly defined by W-Grid, information about peers location (W-Grid coordinates) using as key (both for insertion and search) the peers IDs. In this way, a node (n_s) that need to communicate with another node (n_r) simply searches the network for the ID_r and will discover where n_r can be found. After this, n_s will be able to send a message to the recipient simply using the W-Grid routing algorithm.

In order to show W-Grid capability of managing multi-dimensional data we will define the node ID as a pair (prefix,number) where $Dom_{prefix} = [0, 9999]$ and $Dom_{number} = [0, 9999999]$. We use a hashing function (please refer to [8] and [12] for details) to translate IDs into a binary string of arbitrary length.

For instance, if n_s needs to contact the peer n_r identified by $ID_d = (7601, 452789623)$ it can find[5]:

$$c_d = *10011100$$

$*10011100$ corresponds to the virtual coordinate holding n_d location information, however it is not guaranteed that c_d actually exists in the network. In fact, we estimated a length of 8 bits but, since we work in a distributed environment, we are not able to know in advance the exact depth of the tree structure. Thus the computed string may need to be extended or it can happen that we must stop at a parent coordinate when traveling towards c_d. However, it is not really important which length l is chosen by the sender of the message since at any time any crossed peer can extend[6] the destination string without affecting the correctness of previous steps. Therefore we are sure that every data inserted in the network can be retrieved even with no global knowledge about the network (and implicit W-Grid structure). This location service example is just one of the possible data management applications implementable in W-Grid. In fact, it is possible to manage each kind of one-dimensional or multi-dimensional data by translating them into binary string with the use of hashing algorithms.

V. Experimental Results

To evaluate the performances of W-Grid algorithm we implemented a Network Simulator in Java. We simulated network deployment upon areas with different dimensions, generated nodes in random positions but avoiding partitions in the network. This means that no nodes were isolated.

We particularly focused on virtual coordinates generation and generation of random messages and we also applied the described location service to the simulation in order to evaluate:

- the Average Path Length (APL, measured in hops) covered by messages sent from one node to another, comparing the APL of W-Grid with the one achievable using GPSR;
- the average storage load at each peer, supposing that the information to be managed are virtual locations of nodes, to evaluate the effects of the storage load balancing algorithm.

To achieve that the simulator includes an event manager that triggers periodic beaconing (performed every 300ms) of nodes and generates messages at a parameterizable frequency. The beaconing is asynchronous, namely each peer

[5] By standardizing 7601 and 452789623 to their domains we get 0,76 and 0,45 respectively. We multiply both of them by 2^4 to get a string of length 8. The binary conversion of the multiplications are **1010** and *0110* respectively. Then, by crossing bit by bit the two string we get the c where destination node location is stored $*\mathbf{1}0\mathbf{0}1\mathbf{1}1\mathbf{0}0$.

[6] See [12] for details

TABLE I
RESULTS FOR DIFFERENT AREA DIMENSIONS (50 SIMULATIONS EACH; 50000 MESSAGES SENT)

	APL(in hops)		RMSE		Lost messages	
Area(nodes number)	W-Grid	GPSR	W-Grid	GPSR	W-Grid	GPSR
800×800(120)	6,13	7,49	3,11	8,44	-	2,77%
1000×1000(200)	8,05	9,02	4,45	13,00	-	2,26%
1200×1200(290)	9,75	9,64	4,47	12,74	-	2,01%
1400×1400(400)	11,54	10,87	4,99	14,52	-	3,59%
1600×1600(520)	13,96	13,71	5,86	14,99	-	4,52%
1800×1800(660)	14,81	14,14	6,41	12,15	-	7,88%
2000×2000(820)	17,43	16,57	8,44	13,20	-	8,47%

is independent from the others, as it happens in real networks. For radio transmissions we suppose a range of 100 meters. The manager takes into account the synchronization of tasks on critic resources such as radio device, thus nodes cannot execute different task using the same device at the same time (a node that is beaconing cannot route packets, etc.). For each different task we set appropriate and realistic durations. Coordinate creation is gradual, in fact the simulation randomly choose one node that beacons first and elects itself as root of a new virtual coordinate space. Then, as described in Section III we let periodic beaconing building the W-Grid network. Beside coordinate creation at nodes joining the simulator puts their (randomly) assigned ID into the network as an example of data management and to evaluate to effects of storage load balancing algorithm.

A. Average path length comparison

Once that every node had got its c the simulator (50 simulations per each different area) started the generation of 50000 messages between randomly chosen couples of sender/recipient nodes. Each message was routed according to our algorithm, following the virtual coordinates, and at the same time it was routed using GPSR algorithm (exploiting [x,y] physical positions of nodes).

Even if the comparison appears prohibitive, since GPSR can stay very close to the ideal routing algorithm also because it uses physical position of nodes, W-Grid returns amazing performances, especially considering that it doesn't require any kind of information about geographic position of nodes. This means not only a vaster and heterogeneous space of application, not limited only by GPS (or any other position estimation equipment) embedded devices, but also an easier deployment in every condition and everywhere. Table I shows that the number of hops (APL) is almost the same in W-Grid and GPSR, but if we consider the natural advantage of GPSR that knows physical positions of the nodes we can say that the results are very good since, in some configurations our algorithm presents better performances, due to the perimeter issue of GPSR that can cause longest paths. Besides, it is important to say that W-Grid doesn't fail any message delivery and it performances are almost the same in the different runs per area (see W-Grid MSE) showing that it is not affected by network topology. On the other side GPSR presents a notable percentage of routing failures and its performances are variable and dependent from nodes positions.

B. Load Balancing evaluation

The second aspect we focused on was load balancing at nodes in terms of data managed. Observing our implementation of location service we ran different simulation with and without using our SLOB algorithm. From Figure 5 we can see that its impact is really positive on storage load distribution among peers. We used a bucket size $b = 1$ so that the system aims to achieve a perfect storage load balance with each peer that hold exactly one data.

Figure 5 represents the storage distribution among peers with and without the application of SLOB algorithm. We can clearly see that in simulations where the algorithm is not used the percentage of nodes that store at least one data is less than 10%. Each node of this 10% manages on average 15,04 data and the root mean square error is 24,44. The situation is really unbalanced and the most loaded node can have up to 200 data in worst cases. On the other side, by applying the algorithm we can take up to 90% (about 500 nodes out of 560) the number of nodes that store at least one data. In this case peers manage about 1,14 data each and the root mean square error is 0,36.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel decentralized infrastructure that self-organizes wireless devices in an ad-hoc network, where each node has one or more virtual coordinates through which both message routing and data management occur without reliance on either flooding/broadcasting operations or GPS. The resulting ad-hoc network does not suffer from the dead-end problem, which happens in geographic-based routing when a node is unable to locate a neighbor closer to the destination than itself. The multi-dimensional data management has been described showing, as an example, how a location service reduces to a simple query, like for any other data type. Performance analysis and experiments conducted have showed

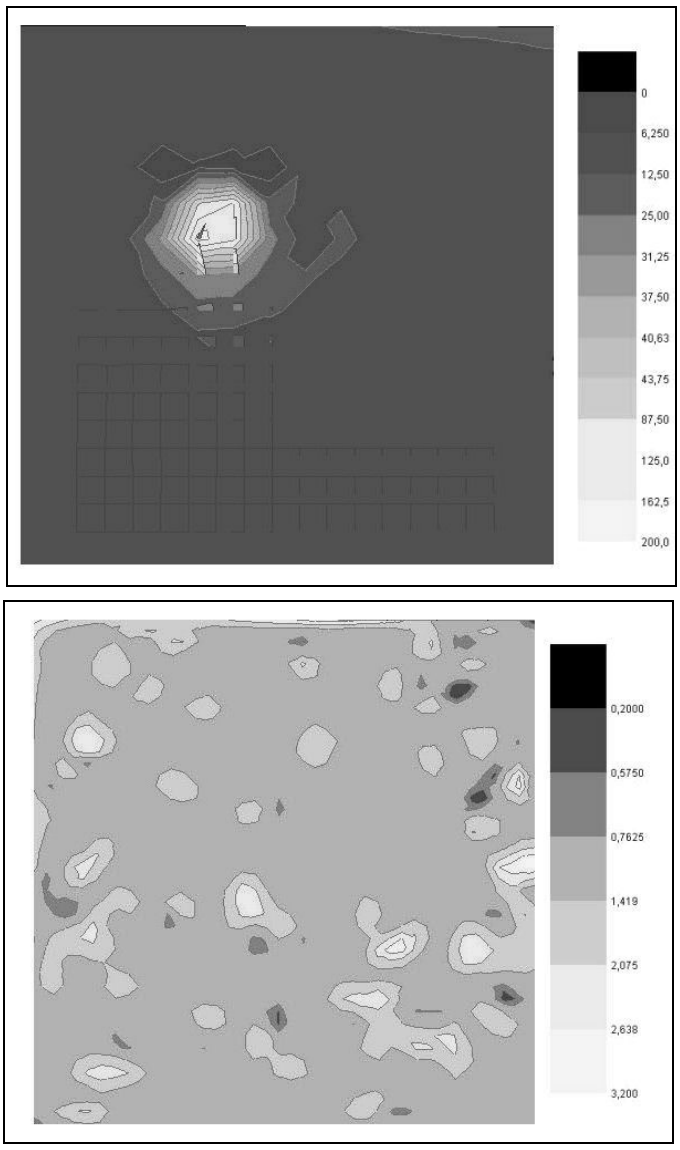

Fig. 5. Contour showing storage load at peers without and with Storage Load Balancing Algorithm

significant performance when compared with GPSR.

Future works will concern the introduction of multiple virtual spaces (namely multiple roots) among which nodes can choose at routing time the next hop according to the space which better reduces the distance to the destination. We are also studying the possibility of introducing a path learning capability at nodes in order to improve the W-Grid APL.

References

[1] R. Bischoff and R. Wattenhofer. Analyzing connectivity-based multi-hop ad-hoc positioning. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 165, Washington, DC, USA, 2004. IEEE Computer Society.

[2] W. L. C. Chiang, H. Wu and M. Gerla. Routing in clustered multihop, mobile wireless networks. In *Proc. IEEE SICON'97*, pages 197–211, April 1997.

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

[4] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[5] D. Johnson, D. Maltz, and J. Broch. Dsr: the dynamic source routing protocol for multihop wireless ad hoc networks. *Ad hoc networking*, pages 139–172, 2001.

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

[7] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72. ACM Press, 2003.

[8] G. Moro, G. Monti, and A. Ouksel. Merging G-Grid P2P systems while preserving their autonomy. In *P2PKM*, 2004.

[9] 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*, 2006.

[10] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual coordinates for ad hoc and sensor networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 8–16, New York, NY, USA, 2004. ACM Press.

[11] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2):183–197, 1996.

[12] A. Ouksel and G. Moro. G-Grid: A class of scalable and self-organizing data structures for multi-dimensional querying and content routing in P2P networks. In *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 and O. Mayer. A robust and efficient spatial data structure: the nested interpolation-based grid file. *Acta Inf.*, 29(4):335–373, 1992.

[14] V. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *INFOCOM '97: Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 1405, Washington, DC, USA, 1997. IEEE Computer Society.

[15] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, NY, USA, 1994. ACM Press.

[16] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90. IEEE Computer Society, 1999.

[17] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM Press, 2003.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–340, 2001.

[20] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.