Hierarchical Geographic Routing for Wireless Ad-Hoc Networks

Luis A. Hernando, Unai Arronategui

I3A, University of Zaragoza
C/ María de Luna 1, Ed. Ada Byron, 50018 Zaragoza, Spain
lahernan@unizar.es, unai@unizar.es

Abstract. Geographic routing is a well established solution for scaling in large wireless ad-hoc networks. A fundamental issue is forwarding packets around voids to reach destinations in networks with sparse topologies. All general known solutions need first to get into a dead end, at link level, to be able afterwards to apply a recovery algorithm. These techniques can lead to very inefficient forwarding paths. We propose a novel general approach, based on light weight connectivity maps, C-Maps, distributed among all the nodes in the network to obtain more efficient and robust paths. The main contribution of our method is the distributed Mercator protocol that builds these maps. Each node in this protocol builds and maintains its own C-Map that summarizes connectivity information of all the network around itself using hierarchical regions. This information is more precise from regions closer to the node. Nodes apply greedy forwarding and face routing to the different hierarchical levels of connectivity information. Better paths are obtained with this behavior. Robustness is guaranteed by every node containing its C-Map. Our analytical and simulation work shows that the map state and the communication overhead grows logarithmically with the size of the network.

Key words: Hierarchical Geographic Routing, Wireless Ad Hoc Networks, Connectivity Maps.

1 Introduction

Network scalability is a fundamental issue in routing protocols for wireless ad-hoc networks. An attractive approach to this problem is geographic routing [2, 7] because no other routing information than the set of neighbours has to be maintained. Also, in dense networks, the forwarding method used, greedy forwarding, is very simple. Besides, cross layer methods allow efficient data-centric applications [11] using geographic routing algorithms.

If the network is not dense enough, packets can find voids that might block greedy forwarding with dead ends and, thus, packets can not reach destinations in a network where connectivity exists. The traditional solution to get out of the dead end and forward around the void is face routing [6], which applies the

Please use the following format when citing this chapter:

Hernando, L. A., Arronategui, U., 2007, in IFIP International Federation for Information Processing, Volume 248, Wireless Sensor and Actor Networks, eds. L. Orozco-Barbosa, Olivares, T., Casado, R., Bermudez, A., (Boston: Springer), pp. 203-214.

"right hand rule" in a planar subgraph of the network. But, all general geographic routing algorithms needs to get first in these dead ends at the link level so that, afterwards, they can apply a recovery algorithm to forward around the void [9]. And, in this way, very inefficient paths can be obtained, mainly when dealing with big voids and, worse, with big and/or frequent concavities at the borders of those voids.

We propose a novel general approach to avoid arriving at dead ends at link level and getting more efficient paths to the destination node. Each node builds and maintains a contextual map, a *C-Map*, containing synthesized information of connectivity of hierarchical regions surrounding the node. These regions are square tiles obtained from simple geographical operations.

We assume that radio ranges could be non uniform, links are bi-directional and nodes are assigned absolute coordinates in the network.

Network connectivity information is folded in region connectivity information at the different levels of the map hierarchy. Finally greedy forwarding and face routing is applied on the resulting region connectivity graph. Using geographic routing at region level we achieve more efficient paths, mainly in sparse configuration with big voids and concavities.

Also, node mobility can easily be included integrating the proposal in [10] with our techniques. Energy considerations are not covered in this paper.

The rest of the paper is organized as follows. Section 2 shows a review of existing and related work. In section 3, the connectivity maps are described. Section 4 presents Mercator, the maps construction protocol. Section 5 presents the hierarchical routing protocol. Section 6 offers theoretical information about costs induced by this method. In section 7, simulation results are shown. And, finally, section 8 offers the conclusions of this work.

2 Related Work

GPSR [4] was the first geographic routing algorithm that tackled the problem of voids and concavities. Messages are forwarded using greedy mode whenever possible towards destination until they encounter a dead end. Then GPSR switches to perimeter mode, the recovery algorithm, forwarding the message with a right-hand rule algorithm over a planarized graph outside the problematic area, then, GPSR switches back to greedy mode. GOAFR family of algorithms [7] also implements face routing over a planarized graph as recovery algorithm, improving GPSR's efficiency. It was proved in [5] that geographic routing protocols (like GPSR or GOAFR) which took the *unit disk graph* assumption do not behave correctly. CLDP [5] provides a planarized graph without relying on the previous assumption by continuously probing all available links, which results in a high cost.

GDSTR [9] was proposed as an alternative: instead of using expensive graph planarization algorithms, it maintains a spanning tree, the *Hull Tree*. Nodes in the Hull Tree keep notion of the area covered by nodes in the branches down to

the leafs. GDSTR's recovery algorithm consists in forwarding messages towards the tree's root until they are outside the problematic area.

All these methods need to get into a dead end to be able to apply a recovery algorithm. Depending on the shape of each void, mainly on its size and concavities existence, very long and inefficient paths may result.

In [3] it is argued that to avoid getting into dead ends is better solution that trying to apply recovery algorithms afterwards. The Distance Upgrading algorithm is developed to obtain better paths. But, this method is proposed only in the context of communications from sensors to a base station. Thus, only one destination is present and all other nodes calculate the distance to this unique destination.

Terminode routing [1] is a hierarchical routing protocol that uses geographic routing for distant destinations and more traditional link state routing in near neighbourhood. Terminode routing has a method called Geographical Map-Based Path Discovery for remote routing. It only operates to get anchored paths at the same level and it is assumed that a density map is already available somewhere outside the network. No distribution technique is presented for the maps. This trend is developed with the use of geographic maps [8] from vehicular navigation systems that are introduced in the geographic routing.

No method has been proposed where the network itself builds an explicit geographic connectivity map to improve substantially routing paths as we do.

3 Connectivity Maps

Connectivity information between different areas in the network is stored into a map, the C-Map. The network is modeled as a hierarchy of square areas defined on a 2D euclidean space. Nodes obtain its localization coordinates by means of an external positioning system like GPS or other localization techniques.

A C-Map stores information about a hierarchy of nested square regions surrounding the owner node's location. The highest level of the map, with the biggest squares, determines the area covered by the whole map. Lower levels keep information about connectivity of small areas near the node, while higher levels keep summarized connectivity information about large and faraway areas.

Hierarchy of network regions. Information in a C-Map is organized into M *Map Levels*, starting from 0, the lowest one, to $M-1$. Each map level is composed of Q non-overlapping squares known as *tiles*, properly sized to fit four tiles of level m into one of level $m+1$. The map is centered in order to balance the amount of information in all directions around the node and to facilitate the composition of information explained later, as shown in Fig. 1. Tiles have to be carefully placed according to the following rules:

- Map level m must be completely nested into map level $m+1$, and thus $Q = 4^n, n \in \mathbb{N}$.
- In each Map level, the tile containing the owner node is known as *central tile* and must be completely surrounded by other tiles.

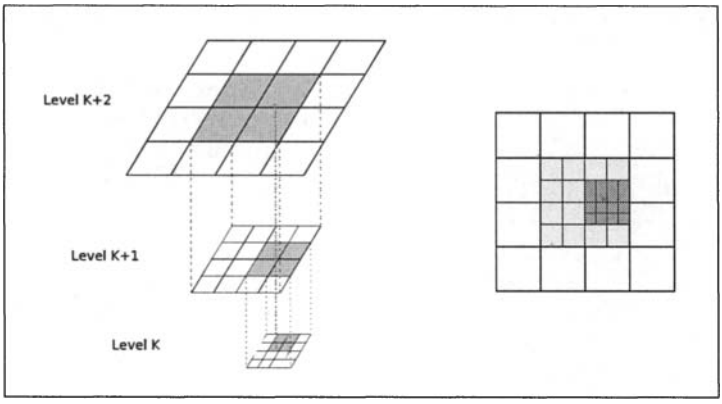


Fig. 1. Hierarchy of nested levels centered around the node owner of the C-Map

Connectivity between regions. Connectivity is defined as the possibility for a message coming from a tile to reach a contiguous one across their common border. Each tile stores the connectivity information of its corresponding square region of the network with its surrounding squares.

Having the connectivity information about all the tiles of each map level, the owner of the map is able to quickly find out if a region of the network is reachable across a path of connected tiles.

Connectivity inside tiles. At first, a tile marked as reachable through all of its borders, would seem like an ideal tile, allowing communications traverse the tile completely, but it is not always true: nodes inside a tile might be divided into different unconnected groups known as *fragments*. This important issue implies that a fragmented tile is not always traversable from one border to other.

A fragment is defined as a connected group of nodes inside a tile and is always traversable.

4 The Mercator Protocol: C-Map Construction

The **Mercator** protocol builds, distributes and maintains the C-Maps for every node in the network in a fully decentralized way. All the information the protocol produces is based on basic connectivity status between nodes. First, connectivity between the smallest tiles is discovered with an interchange of HELLO messages. Then, that information is distributed to the immediate neighbour areas, giving nodes some knowledge about its vicinity and the ability to summarize the information they know. In subsequent iterations of this process, basic and summarized connectivity information will be shared using MAP messages, extending the ability of nodes to build information at higher levels.

After some time, all nodes will be provided with a map of its neighbourhood composed of levels at different scales. Later updates of the map will be required to address the topology changes produced by joining or exiting nodes or even entire network areas.

4.1 Previous Considerations

A square of level m, S_m, is identified by a tuple $<m, (i,j)>$ where (i,j) are S_m's grid coordinates using columns and rows. S_m's side length, L_m, is calculated as $L_m = L_0 * 2^m$, being L_0 a parameter of the network. Given a point P placed inside S_m, (i,j) can be calculated as the integer division of P's coordinates by L_m.

4.2 Mercator Protocol's Information: The C-Map

Information produced by **Mercator** protocol is stored into a C-Map at each node. A C-Map is composed of M Map levels. M is chosen according to the maximum diameter of the network. Each map level is composed of exactly $Q = 16$ tiles arranged in a 4x4 matrix layout (See Fig. 1). The map is 'centered' at the owner node's location, following the rules described in Sec. 3. When a node moves from one tile to another, its map has to be re-centered repositioning the tiles at all levels as needed to meet the previous rules.

As explained before, nodes inside a tile can be divided into fragments. Information stored in the C-Map about each tile comprehends the different fragments the tile is divided into, including for each one a representative identifier unique inside the tile and its connectivity status with neighbour fragments.

4.3 Discovery of Level 0 Information

Once the C-Map is centered, HELLO messages are broadcasted periodically by **Mercator** protocol to gather information about the fragment the node belongs to and its surroundings at the lowest level (level 0). HELLO messages are used also to maintain a fresh list of neighbour nodes.

Each HELLO message contains the following information: sender's node identifier and coordinates, sender's fragment identifier, fragments at neighbour tiles accessible from sender's fragment and the minimum number of hops needed to reach those neighbour fragments from the sender node.

Upon a reception of a HELLO message each node will update its state adding the new information: if HELLO messages are received from a node in a neighbour tile, then the shared border's status among both tiles is considered *active*. The sender's node fragment will be considered accessible from the receiver's fragment and hop count towards sender's fragment will be set to 0. By contrast, if a HELLO message is received from a node which is located in the same tile, then sender and receiver belong to the same fragment. Receiver node will check the HELLO message looking for information updates, i.e. a new active border or a

new link with a fragment, and will keep this information that will be broadcasted in the next HELLO message.

After some stabilization time, each node will be able to compute its fragment's identifier based on which tile's borders seem active.

4.4 C-Map Addition Operation

Connectivity information is shared between nodes exchanging their C-Maps. The previous construction rules of C-Maps imply neighbours' C-Maps must be partially or totally overlapped. A node acquires information from its neigbours' C-Maps applying the addition operation.

Binary addition operator is defined for C-Maps: let C_1, C_2, C_r be C-Maps,

$$C_1 + C_2 \rightarrow C_r$$

where C_r is the combination of C_1 and C_2. C-Map addition operator works as follows: first, C_r is centered at the same point as C_1, then addition operator will select tiles from C_1 not present in C_2 and will copy its information (fragments) into C_r. Then it will select those tiles present in both maps and will add to C_r all fragments from C_1 as well as those from C_2. Finally, if central tiles of corresponding levels in C_1 and C_2 are neighbours and thus share a border, C_2 owner's fragment is marked as connected with C_1 owner's fragment into C_r.

4.5 C-Map Information Exchange

After some stabilization time, each node is able to provide information about its fragment of level 0 and its connections to other neighbour fragments. The node will then start broadcasting MAP messages to its immediate neighbours in order to share information about local and surrounding tiles. Initially, all 16 tiles of each level will be empty except the central tile of level 0, which will include the owner's node fragment.

Each MAP message carries the C-Map stored by the node at the sending time and its owner node's coordinates.

Upon a reception of a MAP message, receiver node will add the received C-Map to its own, acquiring the new information (as explained in 4.4). In the subsequent sendings, the new information will also be spread producing a propagation effect.

4.6 Higher Level Fragment Information Composition

Each node is in charge of the fragments it belongs to at each level and will build and keep them stored into its map. Each node computes from information about fragments of level $m - 1$ which fragment of level m it belongs to. Then MAP messages exchange will propagate fragment's information over the surrounding areas.

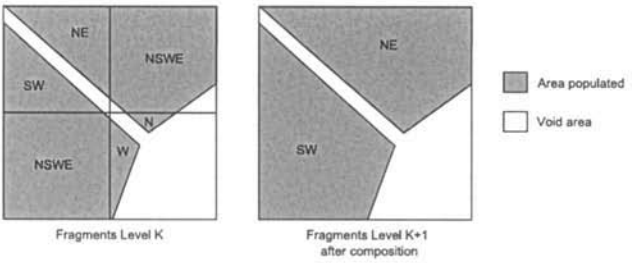


Fig. 2. Composition of a higher level fragment. In the picture, fragments are identified with the cardinal directions of the tile's active borders for the fragment.

Levels in a C-Map are centered at the owner node's location (see 4.2), ensuring that all 16 tiles of level $m-1$ fit perfectly into 4 tiles of level m. The composition algorithm computes node's fragment into its tile of level m, by examining the fragments in tiles of level $m-1$ which fit into its tile of level m (see Fig. 2).

The composition algorithm works as follows: first, it looks for all fragments in the 4 tiles of level $m-1$ which are connected (directly or by intermediate fragments) to the node's fragment at that level. Then, all those connected fragments will conform the node's fragment at level m, calculating also its identifier. Finally, the composed fragment is stored into the level m of the C-Map.

4.7 Information Expiring Mechanisms

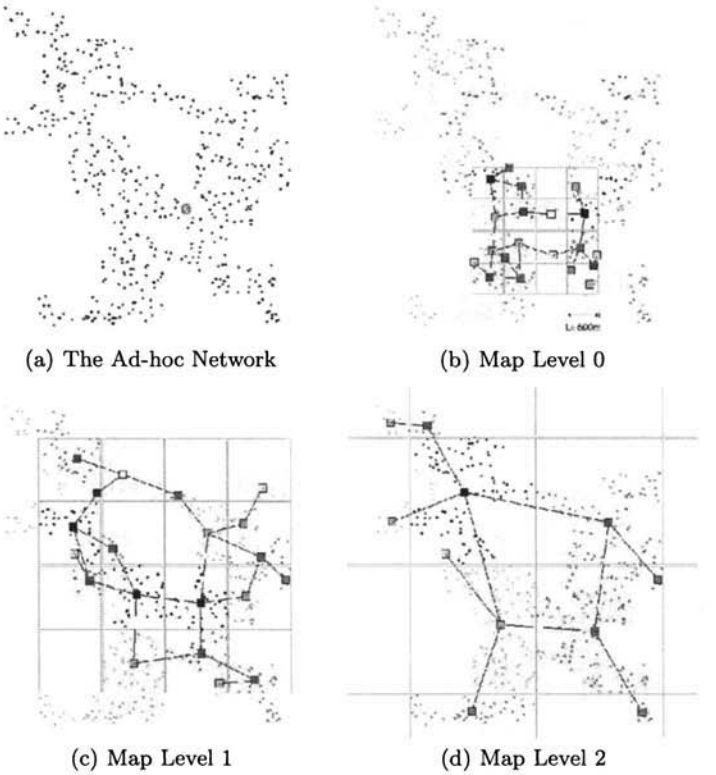
Since ad-hoc networks are not static, nodes appear and disappear creating and removing routes, modifying the network's topology and connecting and disconnecting entire regions. For this reason an invalidation mechanism for obsolete information is proposed.

Every item of a C-Map, like a fragment inside a tile, is associated to an expiration timer that monotonically increases up to a maximum value. Timers information is also stored in the map. An item is removed from the C-Map when its associated timer reaches a predefined maximum value. Nodes in charge of an item produce the information about it, and periodically reset its timer value to 0 like a heartbeat. If an item of a map disappears from the network, i.e. a fragment gains some connectivity and it is renamed, nodes in charge of it will stop resetting the timer for the old identifier and after a period of time it will be removed from all C-Maps where it was contained.

4.8 A Global View of Built C-Maps

The behavior of the **Mercator** protocol on a network of about 500 nodes is shown in Fig. 3(a). It can be seen in Figs. 3(b),3(c) and 3(d) how connectivity

information reflects the shape of the network including void areas and concavities and how a summarized network topology is constructed for each level.



(a) The Ad-hoc Network (b) Map Level 0

(c) Map Level 1 (d) Map Level 2

Fig. 3. C-Map Levels 0, 1 and 2 of the highlighted node in 3(a). Nodes of the same color inside a tile belong to the same fragment, which is represented by a colored square. Connectivity between fragments is represented with straight lines.

5 Hierarchical Geographic Routing with C-Maps

Having the C-Map, traditional geographic routing algorithms can be enhanced to take advantage of the available connectivity information at different scales. Big concavities and voids are very difficult to discover at link level, the lowest level in the map, but its easier to find them on the appropriate higher level.

An improved geographic routing algorithm that takes advantage of C-Maps would be pretty similar to the current ones, performing greedy forwarding through

safe locations in the map, and thus, avoiding problematic areas in order of importance, instead of simply forwarding towards the destination point as in traditional geographic routing.

Enhanced Geographic Routing with C-Maps would work as follows: when a message is forwarded, a node would look in the map for the highest level where destination and current locations are placed inside different tiles of the Map Level. A greedy algorithm applied to the connectivity graph represented at that level will select the next fragment the message should be forwarded to. A waypoint is established at that fragment and the routing protocol will apply greedy forwarding towards that waypoint in the immediate lower level of the map. This process is repeated until Level 0, resulting in a direction suitable for greedy routing at link level. Known recovery algorithms would work in a similar way but applied to the graph at the level which encounters trouble.

With this approach problematic areas are avoided at all levels reducing to the minimum the usage of recovery algorithms.

6 Costs Estimation

The **Mercator** protocol has been designed in order to achieve high scalability properties with target networks from a few to hundred thousand nodes. The design has been focused on keeping conservative storage requirements per node and shared channel bandwidth usage.

C-Map storage cost. A look to the data structure managed by **Mercator** Protocol, the C-Map, shows the powerful scalability properties it offers. Each node in the network stores only its own C-Map which is composed of M Map Levels. Each Map Level stores $Q = 16$ tiles which might be divided into different fragments. Storage cost of a C-Map, S_{Cmap} can be calculated as follows:

$$S_{Cmap} = M * Q * F * f \tag{1}$$

Where F is the average number of fragments per tile and f is the storage cost of a fragment. S_{Cmap} increases linearly with M.

The central tile of each level is surrounded by at least one tile on each direction (See Sec. 3). This property guarantees for a Map Level K, a complete coverage of a network of L_K meters of diameter in a worst case scenario, where the owner node is in the farthest possible location from the middle point of the Map Level. The maximum coverage area of a C-Map, A_{Cmap}, is determined by its highest Map Level, in fact, M is chosen to guarantee coverage for a maximum network diameter. A_{Cmap} can be calculated as follows:

$$A_{Cmap} = (L_0 * \sqrt{Q} * 2^{M-1})^2 \tag{2}$$

Table 1 shows the coverage areas of C-Maps with different number of levels in a default setup, being $Q = 16$ and $L_0 = 500$ meters. A_{Cmap} increases exponentially with M.

M:	1	2	3	4	5	6	7	8
$A_{Cmap}(Km^2)$:	4	16	64	256	1024	4096	16384	65536

Table 1. Coverage area of a C-Map with M map levels

Mercator bandwidth usage. All **Mercator** information exchange is performed by means of local broadcast operations. Once the sender node transmits a message, it is not forwarded again by any of the receiver nodes.

Connectivity information discovery and distribution is done by means of periodic HELLO and MAP messages sendings. HELLO and MAP messages sending rate is calculated in order to use a low percent of the available channel bandwidth. Shared medium bandwidth used by *Mercator* in nodes surrounding point P, $M_{Bw}(P)$, can be approximated as follows:

$$M_{Bw}(P) = D(P) * A_{RR}(P) * (R_{HELLO} * S_{HELLO} + R_{MAP} * S_{MAP}) \quad (3)$$

Where, $D(P)$ is the average network population at point P, $A_{RR}(P)$ is the radio range area of a node in P, R_{HELLO} and R_{MAP} are the broadcast rates for HELLO and MAP messages, S_{HELLO} and S_{MAP} are the average size of HELLO and MAP messages and C_{Bw} is the channel bandwidth.

Wireless communications use a shared medium, thus, node density will affect the bandwidth employed by **Mercator** at any point in the network. We propose two strategies for dealing with node density:

1. Sending rates depend on D. For instance, taking $R_{MAP}(D) = K/D$, where K is a predefined constant, then, $R_{MAP}(D)$ will decrease as node density grows. This approach would help to reduce the channel bandwidth used by **Mercator**, turning M_{Bw} not dependent on D. Each node approximates D as the number of neighbours $+1$ divided by its area of radio coverage. This approach will produce an approximately constant bandwidth usage per square meter not dependent on D.
2. Use a notification mechanism instead of expiration timers. This mechanism, upon an inconsistency between information at a level with its immediate upper level would recalculate connectivity information for the upper level and spread it.

7 Experimental Results

We have implemented an ad-hoc network simulator to test the **Mercator** protocol under different scenarios. The following settings have been used in our tests: radio range is 200 meters. All nodes receive broadcasted messages within their radio range. R_{HELLO} is set to 1 message per second. R_{MAP} is set to 1/3 messages per second. Expiration ages are set to 4 *ticks* for each item. Tick duration is $1/R_{MAP} * 2^n$ seconds where n is the level number of the item monitored by the timer.

The side's length of a tile at level 0, L_0, is closely related to expiration ages and radio range. L_0 value has been set to 500 meters, which allows information to traverse an entire tile of level 0 in less than 4 hops without expiring.

Storage and bandwidth requirements: with the above parameters, a C-Map with 9 levels (262,000 square kilometers of coverage area and average network population of 2,600,000 nodes) requires 5.3KBytes of storage (according to (1)). In this scenario, an average neighbour count of 4 produces an approximated channel bandwidth usage of 71kbps (according to (3)).

Stabilization time. In a first test we have measured the map's stabilization time, which provides an idea of the response speed that the protocol offers against network topology changes. Stabilization time is measured by the number of MAP messages each node sends until the entire network map is completely built. We have simulated a cold startup in a network with 7000 nodes under different population density conditions (from 4 to 9 average neighbour count). As shown in Fig. 4, stabilization time increases linearly with the extension of the mapped area, and thus, exponentially with the number of levels of the C-Map. High density conditions help to increase information propagation speed and to reduce stabilization time.

We have also simulated dynamic scenarios with nodes joining and going away from the network. If a new node does not produce a substantial change into the network topology, stabilization time is about 1 message, by contrast, if a new link is created between two fragments, the stabilization time is similar to the case of a cold startup at the level of those fragments.

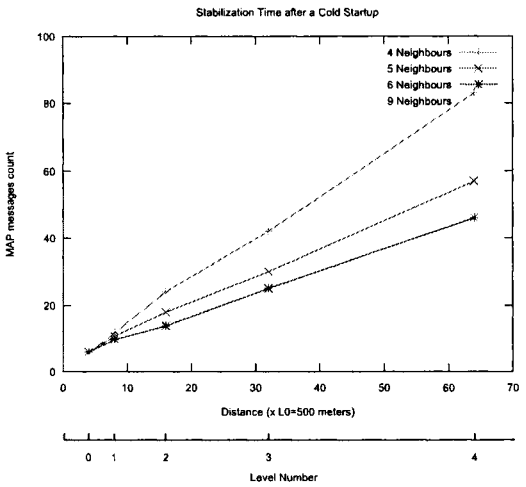


Fig. 4. Stabilization time after a cold startup

8 Conclusions

In this paper we propose a new approach in geographic routing that improves considerably routing paths in sparse networks with big voids and/or significant concavities. More efficient paths than classical routing and recovery algorithm can be obtained. Connectivity robustness is guaranteed with map distribution among all nodes. It can marry nicely with existing mobility solutions. Ongoing work explores the use of the C-Maps with other routing methods than traditional geographic routing, with same lightness considerations, to achieve even better routes.

Acknowledgments. This work has been supported by the Spanish CICYT DPI2006-15390 project and the GISED, group of excellence recognised by the Diputación General de Aragón.

References

1. L. Blazevic, J. Le Boudec, S. Giordano: A Location Based Routing Method for Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing **4**(2) (2005) 97–110
2. P. Bose, P. Morin, I. Stojmenovic, J. Urrutia: Routing with guaranteed delivery in ad hoc wireless networks. Wireless Networks **7**(6) (2001) 609–616
3. S. Chen, G. Fan, J. Cui: Avoid "Void" in Geographic Routing for Data Aggregation in Sensor Networks. International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Special Issue on Wireless Sensor Networks, **1**(4) (2006) 169–178
4. B. Karp, H. T. Kung: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In Proceedings of the 6th ACM International on Mobile Computing and Networking (MobiCom 00) (2000) 243–254
5. Y. J. Kim, R. Govindan, B. Karp, S. Shenker: Geographic Routing Made Practical. In Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (2005)
6. E. Kranakis, H. Singh, and J. Urrutia: Compass routing on geometric networks. In Proceedings of the 11th Canadian Conference on Computational Geometry (1999) 51–54
7. F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger: Geometric ad-hoc routing: Of theory and practice. In Proceedings of PODC (2003) 63–72
8. M. O. Legner: Map-based Geographic Forwarding in Vehicular Networks. University of Stuttgart, Faculty of Computer Science, Diploma Thesis No. 1994 (2002)
9. B. Leong, B. Liskov, R. Morris: Geographic Routing without Planarization. In Proceedings of the 3rd Symposium on Network Systems Design and Implementation (NSDI 2006) (2006)
10. M. Li, W.-C. Lee, A. Sivasubramaniam: Efficient peer-to-peer information sharing over mobile ad hoc networks. In Proceedings of the 2nd Workshop on Emerging Applications for Wireless and Mobile Access (MobEA 2004) (2004)
11. S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, F. Yu: Data-centric storage in sensornets with GHT, a geographic hash table. Mobile Networks and Applications **8**(4) (2003) 427–442