Mercator: Self-Organizing Geographic Connectivity Maps for Scalable Ad-Hoc Routing

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. A fundamental problem of future networks is to get fully self-organized routing protocols with good scalability properties that produce good paths in a wide range of network densities. Current approaches, geographic routing and table based routing, fail to provide very good scalability with good paths in sparse networks. We propose a method based on the discovery of connectivity between geographic regions that are self-organized in a multilevel hierarchy. The Mercator protocol builds lightweight connectivity maps in a fully decentralized manner and shows a scalable and resilient behaviour. Each node builds and maintains its own hierarchical map that summarizes connectivity information of all the network around itself using geographic regions. Link state routing is used over the multilevel connectivity graph of the map to obtain global paths. The analysis and simulation of our approach shows that routing state and communication overhead grows logarithmically with network size while producing good paths.

Key words: Self-Organizing Hierarchical Protocol, Geographic Connectivity Maps, Link State, Scalable Ad-hoc Routing.

1 Introduction

A fundamental problem of future ad-hoc networks is to get self-organized routing protocols with good scalability properties and good enough paths. On the one hand, best scalability properties are obtained with geographic routing [9, 3, 10] where in the best case, there is no communication overhead, location service aside, and routing state is only related to one hop nodes. But, this kind of protocols gives efficient paths only with regular dense networks. On the other hand, table-based routing achieves best paths in all kind of networks, but some state and communication overhead is always needed, even with best ad-hoc hierarchical protocols [14]. Also, scalability with sufficient resiliency seems another challenge.

We propose a self-organizing method that achieves less state than ad-hoc routing protocols based on tables, and better paths than geographic routing protocols. Also, its assures resiliency. In our method, routing is done through

multilevel geographic regions. All nodes have an identifier and an address based on GPS like absolute geographic coordinates or derived from GPS enabled neighbours [6]. These coordinates are used like node addresses and region identifiers. The Mercator protocol builds connectivity maps, C-Maps, based on geographic regions in a fully decentralized way. In this protocol, each node builds and maintains its own C-Map containing a summary of connectivity information from hierarchical regions around the node. A fisheye approach on the maps gives more detailed information in the proximity of the node. These regions are square tiles obtained from simple geographic operations. Link level connectivity information is processed and folded in region connectivity information with a recursive technique. Link state routing is applied to the graphs that represent connectivity between the regions, at different levels, from the C-Maps. With Mercator, we achieve routing state that only depends on the number of levels in the map hierarchy. Routing communication overhead is severely reduced because low level updates can remain inside a region scope, if enough density is provided. The number of levels of the hierarchy arises logarithmically from node density in regions.

The proposal in this paper takes a conservative approach with size of C-Maps to minimize stored state and traffic loads. Actually, our results in routing state and communication overhead show that even sensor devices, as the MICAz [5] with 128k of memory for programs and 256kb/s of bandwidth, can be candidates to adopt our approach. Of course, energy consumption has to be considered in sensor networks, which is out of the scope of this paper.

However, in networks with more resources, like mesh networks or vehicular networks, the detail of a C-Map can be increased to obtain better routes. A trade-off between route lengths and overhead needs to be evaluated in each network class to define the size of C-Maps.

Also, location services and node mobility can be easily included, integrating scalable proposals like [12]and [13] with our techniques.

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 routing protocol. Section 6 offers theoretical information and experimental results about costs induced by the protocol. And, finally, section 7 offers the conclusions of this work.

2 Related Work

Two main approaches have been adopted in order to achieve good scalability properties in ad-hoc networks. On the one hand, hierarchical routing protocols divide the routing problem in different spatial or temporal levels. On the other hand, geographic routing protocols use the physical network coordinates of nodes to perform position based forwarding.

Implicit hierarchy is observed in flat protocols like Fisheye State Routing (FSR) [15] and Hazy Sighted Link State (HSLS) [16]. FSR provides accurate

path information from the immediate neighbourhood of a node and imprecise knowledge of paths to distant destinations. Although, this imprecision is solved with the route being more accurate as the packet draws closer to the destination. We have been inspired by this approach to limit the communication overhead while maintaining robustness. Both protocols achieve low communication overheads by selectively adjusting frequencies of routing updates. However, classical heavy O(N) routing state is needed in both protocols which could be a fundamental limitation in scalability.

Hierarchical State Routing (HSR) [14] is a multilevel clustering-based link state routing protocol. A clustering scheme recursively defines a logical hierarchical topology. Communication and storage complexity are the best, H*K, growing linearly with hierarchy levels (H) and average one hop neighbours of a node (K). But, resiliency problems can arise as there is only one clusterhead per cluster to link the hierarchy. Also, cluster needs to be assigned a hierarchical address to aggregate in the node address format. ARCH [1] decrease the cost of the organization of clusterheads avoiding the rippling effect. But, one clusterhead per cluster is still a bottleneck.

In Zone-Based Hierarchical Link State (ZHLS) [8], a GPS-based method is used to define scope of the regions without clusterheads. A peer-to-peer technique reduces traffic bottleneck, avoids single point failures, and makes mobility management easier. There are only two levels in the hierarchy. We have generalized this approach with a multilevel hierarchy and we have reduced the routing state applying a fisheye technique. The GAF [17] protocol also uses a GPS partitioning scheme of the network in square grids but it's focused exclusively on energy conservation and can run over any ad-hoc routing protocol.

Although geographic routing is simple and light, it has important routing problems with sparse networks where voids can block position based forwarding. This dead ends are avoided with recovery algorithms. GPSR [9] uses a right hand rule algorithm over a planarized graph to get around the void. However, voids with complex and big shapes can lead to very inefficient paths. So far, GDSTR [11] is the best recovery solution that improves path lengths. It is based on a spanning tree built on convex hulls. But, it also applies the recovery algorithm after getting in a dead end. Thus, it is sensitive to voids and concavities in the network in order to get shorter routes.

Terminode routing [2] and GeoLANMAR [18] are hybrid protocols with two levels that use geographic routing for distant destinations and link state in local scope. GeoLANMAR uses classical LANMAR methods to manage logical groups in group mobility applications. Terminode routing has a method called Geographical Map-Based Path Discovery for remote routing. This method 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. Besides, no distribution technique is presented for the maps. This trend is developed with the use of geographic maps [4], from vehicular navigation systems, in geographic routing. However, good node density is assumed to assure connectivity.

Our goal has been to obtain a routing method with better routes than light state geographic routing protocols and smaller routing state than hierarchical routing protocols with managed communication overhead. All this is done assuring resiliency.

3 Connectivity Maps

Connectivity information between different areas in the network is stored into a map, the C-Map. In this approach, the network is modelled as a hierarchy of square areas defined on a 2D euclidean space where nodes are located. Each node is able to 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 stored in a C-Map is organized into M hierarchical *Map Levels*, starting from 0 (the link level), 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 tile of level $m+1$. In addition, C-Maps are centred around their owner node, which means that all tiles of each level are placed around the node. The purposes of centring maps are:

- To balance the amount information stored in all directions around the node.
- To enable the propagation of information, by means of the addition of C-Maps, as explained later.
- To make possible the summarization of information from lower levels into higher ones, using a composition operator. The final goal is to ensure the scalability of the protocol by cutting down the amount of information stored into each C-Map.

C-Maps are constructed and centred following the next two **rules**:

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

In the end, the node owner of the C-Map will be placed into one of the 4 tiles in the middle of each level. Whenever the node moves from one tile to another, its map has to be re-centred by repositioning the tiles at all levels as needed to meet the previous rules.

Connectivity Between Tiles. Connectivity is defined as the possibility for a message coming from a tile to reach a contiguous one across their common border. For each tile in the C-Map, the connectivity information with its surrounding tiles is stored, and as a result, any node having the connectivity information about all the tiles of each map level is able to quickly find out if a region of the network is reachable across a path of connected tiles.

Connectivity Inside Tiles: Fragments. At first, a tile identified as reachable through all of its borders, would seem like an ideal tile, enabling communications to 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

In this paper we propose **Mercator**, a fully distributed protocol that builds, distributes and maintains the C-Maps for every node in the network. All the information the protocol produces is inferred from basic status of connectivity between nodes at link level. 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, every node 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** is stored into a C-Map at each node. C-Maps are composed of a fixed number of Map levels, M, each one filled with

Q tiles arranged in a square matrix layout following the basic centring rules described in Sec. 3. The minimum Q value that meets the requirements explained in Sec. 3 is $Q = 16$ which is adopted by default. Q values lower than 16, do not guarantee that the central tile at any level is completely surrounded by other tiles (i.e $Q = 4$) or do not guarantee the statement $Q = 4^n, n \in \mathbb{N}$, which is a requirement for the addition and the composition operators.

C-Maps are sized in order to cover the whole space occupied by the network, thus, the number of levels stored into the Map, M, is a well known parameter dependent of the maximum diameter of the network. The 4 tiles in the middle of the highest Map Level $(M-1)$ should cover the entire network. The coverage area of a C-Map, A_{Cmap}, can be calculated as follows:

$$A_{Cmap} = {D_{Cmap}}^2 = 2 * {L_{M-1}}^2 = (2 * L_0 * 2^{M-1})^2 \quad (1)$$

Where $D_{Cmap} =$ is the guaranteed maximum diameter of the network covered by the highest Map Level.

Table 1 shows the coverage areas and maximum network diameters achieved by a C-Map with M levels in a default setup, being $Q = 16$ and $L_0 = 500$ metres. Both A_{Cmap} and D_{Cmap} increase exponentially with M.

$M:$	4	5	6	7	8	9	10	11
$D_{Cmap}(Km):$	8	16	32	64	128	256	512	1,024
$A_{Cmap}(Km^2):$	64	256	1,024	4,096	16,384	65,536	262,144	1,048,576

Table 1. Maximum network diameter and coverage area achieved by a C-Map with M map levels in a worst case scenario

As explained before, nodes inside a tile can be divided into fragments. Information stored into the C-Map about each tile comprehends the identifiers of the different fragments that the tile is divided into as well as their connectivity status with neighbour fragments and some expiration timers needed for the management of the map.

4.3 Calculation of Fragment's Identifier

Fragments are separated groups of interconnected nodes within the limits of a tile, and thus, each fragment can only be connected with other ones inside adjacent tiles. The identifier of a fragment reflects its connectivity properties: borders of every tile are associated with *cardinal directions* and, in addition, identifiers of fragments are the aggregation of those cardinal directions of the borders traversed by links with neighbour fragments. If two fragments inside a tile have similar connectivity properties their identifiers will be similar, but they will occupy different tiles at lower levels which will differentiate them. If two fragments have similar identifier at level 0, the geographic coordinates of any

node which belongs to them will make the difference, since a node only belongs to one fragment per level.

4.4 Discovery of Level 0 Information

Once the C-Map is centred, HELLO messages are broadcasted[1] 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 carries information about the sender, its coordinates and level 0 fragment identifier, in addition, it also contains the identifiers of the tile's borders through which link to neighbour fragments are established.

Upon the reception of a HELLO message, each node will update its state: if HELLO messages are received from a node in a neighbour tile, the shared border among both tiles is considered traversed by a link and the sender's fragment will be considered connected to the receiver's fragment. By contrast, if a HELLO message is received from a node which is located in the same tile, then sender and receiver nodes belong to the same fragment and the receiver node will check the HELLO message looking for information about the links from its own fragment to other ones.

The updated information will be broadcasted in subsequent HELLO messages and after some stabilization time, each node will be able to compute its fragment's identifier based on the tile's borders that seem active.

4.5 C-Map Addition Operation

Connectivity information is shared between nodes when they exchange their C-Maps. C-Map construction and centring rules guarantee that the C-Maps of two nodes placed into two adjacent tiles, must be partially or totally overlapped depending on the tile they are placed at each level (See Fig. 1). A node acquires information from its neighbours' C-Maps by applying the addition operator defined as follows:

Let C_1, C_2, C_r be C-Maps,

$$C_1 + C_2 \rightarrow C_r$$

where C_r is the result of the addition $C_1 + C_2$. C-Map addition operator works as follows: first, C_r is centred 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.

[1] Local Broadcast. Broadcasted messages are only received by nodes in range and not forwarded again by any of them.

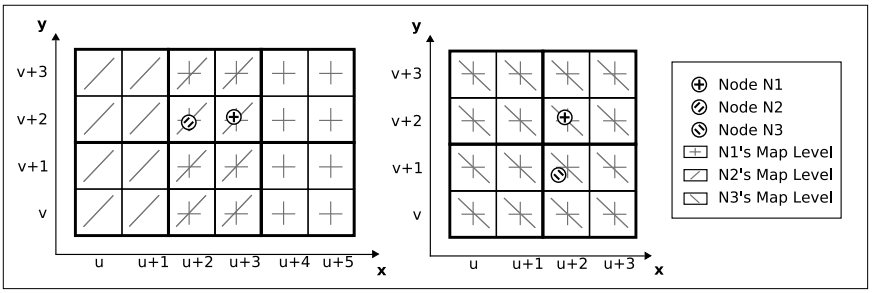

Fig. 1. Partially overlapped Map Levels in the picture on the left: the two central columns are overlapped. Fully overlapped Map Levels in the picture on the right. Note that a node is always placed inside one of the four tiles in the middle of its map level.

4.6 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 and will start broadcasting[2] MAP messages to its immediate neighbours, sharing information about its 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 the reception of a MAP message, the receiver node will add the received C-Map to its own, acquiring the new information (as explained in 4.5). In subsequent sendings, the new information will be spread producing a propagation effect.

4.7 Higher Level Fragment Information Composition

Each node is in charge of the fragments it belongs to at each level and is able to calculate their identifiers and keep them stored into its C-Map. The calculation of identifiers of fragments with information about lower levels is known as composition. As result, information about fragments of 4 tiles at level $m-1$ is summarized into one tile at level m. Internal connectivity details are hidden and external ones summarized.

Each node is able to compute the identifier of the fragment it belongs to at level m because it keeps stored into its C-Map the fragments of level $m-1$ connected to its own. They all conform the fragment of level m. This is possible because levels in a C-Map are centred at the owner node's location (see 4.2). 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 level $m-1$ inside the tile of level m. Then, all those

[2] Local Broadcast, as in HELLO messages broadcasting.

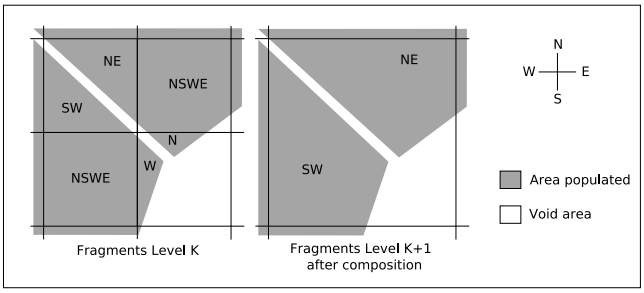

Fig. 2. Example of fragment composition.

connected fragments will conform the node's fragment at level m and its identifier is inferred from their connectivity properties. See Fig. 2.

4.8 A Global View of Built C-Maps

The behaviour of the **Mercator** protocol on a network of about 500 nodes is shown in Fig. 3. It can be observed 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.

5 Routing with C-Maps

Having a C-Map, traditional ad-hoc routing algorithms can take advantage of the available connectivity information at different levels. Implicit avoidance of big concavities and empty areas, which is a big trouble in geographic routing, can be easily achieved by routing with the aid of high level information.

In this section we propose a routing technique inspired in traditional routing algorithms over a network graph. Information from the C-Map is used to construct a graph representing the fragments at all levels in the map and their interconnections.

Two differentiated protocols are involved in the routing process. The first one, known as *Fragment Routing*, acts as a high level path planner, which analyses the previously constructed network graph, looking for the best path through connected fragments of any level toward a target fragment of level 0. The second one, is a classical ad-hoc routing algorithm deployed inside every fragment of level 0 in the network, which is able to deliver a message to a particular node within the same fragment as well as to forward the packet to a node connected with a neighbour fragment.

In Fig. 3, a message is sent from *source node* toward *destination node*. Different snapshots have been taken at different steps: just before the message is sent by source node and by the time it passed through two intermediate nodes. It can be observed how the selected best path is progressively more accurate as the message gets closer to destination.

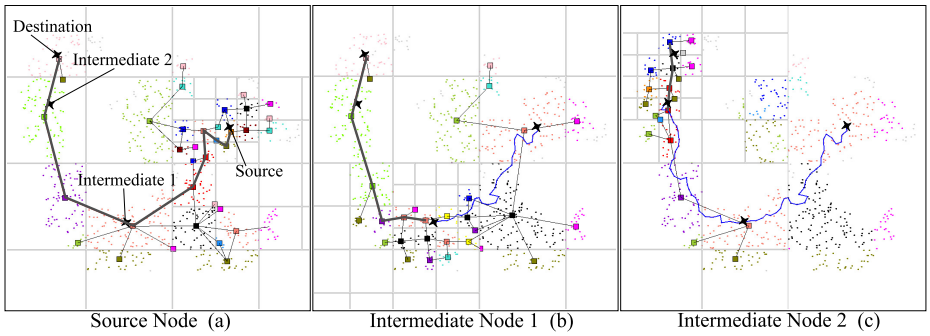

Fig. 3. A message is sent from *source node* to *destination node*. Snapshots at different moments are taken showing the network connectivity graphs up to level 2, which are known by Source Node in Fig. 3(a), by Intermediate Node 1 in Fig. 3(b), and by Intermediate Node 2 in Fig. 3(c). The planned best path in each snapshot, from current node toward destination, is displayed using the thickest grey lines. The winding line represents the exact link level path already followed by the message.

Mercator Node Addressing. Node addressing in the ad-hoc network is designed to provide fragment information available at all levels. Mercator node addresses contain the addressee's geographic coordinates and the identifier of all the fragments it belongs to (one per level). Since coordinates of tiles can be calculated from the coordinates of a point inside it, (see Sec. 4.1), localisations of fragments referred in the address are well known.

A *Mercator address* is in the form: $<(x, y), f_0, f_1, \ldots, f_{M-1}>$, where x, y are the addressee's coordinates and f_k are the identifiers of the fragments it belongs to at each level k. The size of a Mercator address, $W_{address}$, can be calculated as follows:

$$W_{address} = 2 * \left\lceil \log_2(L_0 * \sqrt{Q} * 2^{M-1}) \right\rceil + M * W_{id} \qquad (2)$$

Using 1 metre precision in 2D integer coordinates (x, y), and where W_{id} is the size of a fragment's identifier (4 bits). Table 2 shows the address length required for different network diameters in a default setup, with $Q = 16$ and $L_0 = 500$ metres.

M :	6	9	12	15	17
$D_{Cmap}(Km)$:	32	256	2048	16384	65536
$W_{address}(bits)$:	54	72	90	108	120

Table 2. Network diameter covered by networks with M levels and the corresponding Mercator address length.

The Network Graph. Information stored into a node's C-Map is enough to calculate a network graph with the summarized network topology information. Each fragment is represented by a vertex. A link between two fragments is represented by a link between their corresponding vertexes while the weight of each link represents the approximated cost of routing a message from a fragment to the other and is set to the distance between the central points of the two tiles where the fragments are placed.

It is remarkable that the graph represent fragments (not nodes) which belong to different levels. The topology reflected in the graph is more detailed in areas closer to the node due to the particular construction of the C-Map. Since the number of tiles that a map stores is fixed and the number of fragments stored per tile is bounded, the cost of processing a graph is upper bounded.

Fragment Routing. This routing algorithm, which is run on every node, computes the best path over the network graph, that starting from the node's fragment at level 0, reaches the destination node's fragment at the lowest possible level. The next hop in the path is then selected and the message is forwarded toward it. A routing table at each node is periodically constructed, with entries for all available fragments from node's C-Map. The routing table is calculated applying Dijkstra's Algorithm to the network graph. Every possible destination fragment in the graph is associated with a next hop, the beginning of the shortest path to that destination, which is, a fragment of level 0 and a neighbor of the node's fragment.

At the time of routing a message, the protocol will use the Mercator address of the message's destination, to choose the best next hop fragment. A search is performed in the routing table, looking for a fragment that matches in coordinates and identifier to any of the included in the address. If more than one match is found (at different levels), the one with lowest cost is chosen.

Ad-Hoc Routing Inside Fragments of Level 0. A traditional ad-hoc routing protocol is deployed inside each fragment of level 0 with two objectives. First, once *Fragment Routing* has selected the next fragment-hop toward destination, this ad-hoc routing protocol takes charge of the message and forwards it to the nearest node within the fragment which is directly connected to the next hop fragment. And second, when a message finally arrives to the fragment of level 0 which contains the destination node, the protocol is in charge of the delivery of the message to its final destination.

Understanding How Routing with C-Maps Works. C-Maps are centred, which means that only a part of the network is mapped at one level, so, if a level does not cover a destination target, a higher map level has to be used because it contains information less accurate but broader.

As a message is being routed and approaches its destination, C-Map information into intermediate routers gets progressively more accurate toward the destination target and the path can be slightly modified and optimized.

6 Costs Analysis and Experimental Results

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

C-Map Storage Complexity. 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:

$$S_{Cmap} = M * Q * F * f \tag{3}$$

Where F is the average number of fragments per tile and f is the storage cost of a fragment (fixed). Although F is not a fixed value, our simulations show that only in very specific cases it grows over 2.0. Thus, storage complexity is $O(M)$ and grows logarithmically with network size.

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 broadcastings. HELLO and MAP messages sending rate is calculated in order to use a low percent of the available channel bandwidth. In the simplest scenario, without using any kind of optimization like data compression or variable sending rates, bandwidth required by *Mercator* per node, M_{Bw}, can be calculated as follows:

$$M_{Bw} = R_{HELLO} * S_{HELLO} + R_{MAP} * S_{MAP} \tag{4}$$

Where R_{HELLO} and R_{MAP} are the broadcast rates for HELLO and MAP messages and S_{HELLO} and S_{MAP} are the average size of HELLO and MAP messages. All parameters but S_{MAP} are constant, S_{MAP} complexity is $O(M)$ because each MAP message contains a C-Map. Thus M_{Bw} complexity is $O(M)$.

6.1 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 metres. All nodes receive broadcasted messages within their radio range. R_{HELLO} is set to 1 message per second. R_{MAP} is set to 1 message each 3 seconds. 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. 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 metres, which allows information to traverse an entire tile of level 0 in less than 4 hops without expiring.

Storage and Bandwidth Requirements. Using the above parameters, a C-Map with 9 levels (area of target networks up to 65,000 square kilometres and average network population of 1300 million nodes using a density of 1 node per 200 m^2) requires 5.3KBytes of storage (according to Equation (3)). In this scenario, the bandwidth usage per node is 14kbps (according to Equation (4)).

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 that 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. High density conditions help to increase information propagation speed and to reduce stabilization time.

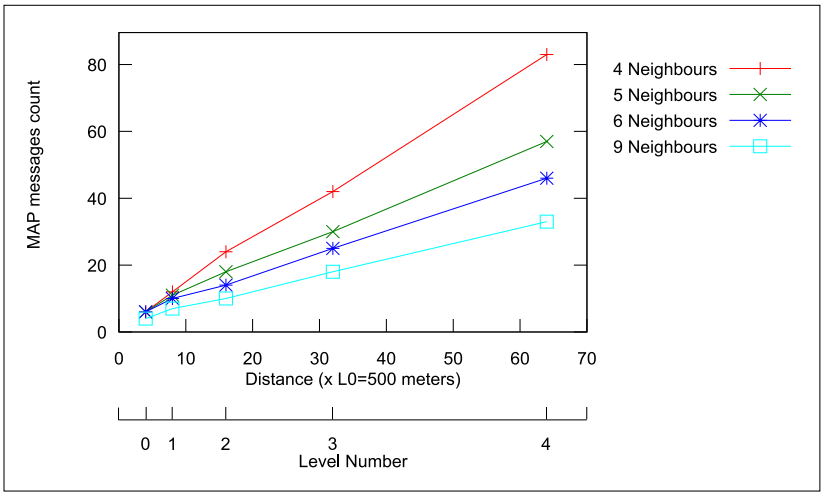

Fig. 4. Stabilization time after a cold startup

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 of the level of those fragments.

6.2 Observations

Storage required by Mercator protocol grows logarithmically with the network area, maintaining a very reduced map with the network connectivity status.

A C-map fits easily into today's sensors memory [5] even for very large target networks with thousand kilometres of diameter.

The storage cost reduction is made possible by the hierarchical approach of the map. Our proposal gives a generalized algorithm for M levels so it can be applied to target networks of very different natures.

A consequence of the low storage cost is that complete replication is feasible. There are no central nodes in the network: every node in the network keeps a map, which improves network resilience, avoids network congestion around critical points and helps with adaptive deployment: a node joining to a previously established network needs only one map from its immediate neighbour to get all the information it needs to become communicated.

At the highest levels, the fact that connectivity status information is summarized and the big size of network regions compared to link level radio range, turn the probabilities that a change at link level produce a change in the connectivity information of high levels very low. Information summarization helps to keep hidden into lower levels the vast majority of connectivity changes, leaving higher levels very stable. This fact greatly simplify node mobility management.

Destination addresses of moving nodes must change as they move from one region to another. Since nodes move with a low speed in relation to network diameter, in most cases, movements will produce a minor change into the address. This fact can be exploited by a distributed location service for efficiency improvement.

7 Conclusions

In this paper we propose a new approach in ad-hoc routing based on nodes with absolute geographic coordinates. The Mercator protocol uses these coordinates to build a multilevel connectivity C-Map summarizing network connectivity information with a fisheye approach. Resiliency is guaranteed with C-Map computation and distribution done in all nodes. It can nicely marry with existing mobility solutions. Routing state and communication overhead grow logarithmically with network size. Better paths than light state geographic routing can be obtained and smaller routing state than hierarchical routing protocols is provided.

Future work will explore the extension of C-Maps with information about mobility and congestion.

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. E. M. Belding-Royer: Multi-level Hierarchies for Scalable Ad hoc Routing. Wireless Networks (WINET) **9**(5) (2003) 461–478

2. 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
3. P. Bose, P. Morin, I. Stojmenovic, J. Urrutia: Routing with guaranteed delivery in ad hoc wireless networks. Wireless Networks **7**(6) (2001) 609–616
4. A.M.K. Cheng, K. Rajan A digital map/GPS based routing and addressing scheme for wireless ad-hoc networks. Proceedings of IEEE Intelligent Vehicles Symposium (2003) 17–20
5. Crossbow Technology, Inc.: MICAz wireless modules. http://www.xbow.com/Products/productdetails.aspx?sid=164 (2007)
6. T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher: Range-Free Localization Schemes in Large-Scale Sensor Networks. In Proc. of the 9th Intl. Conference on Mobile Computing and Networking (MOBICOM) (2003) 81–95
7. Xiaoyan Hong, Kaixin Xu, and Mario Gerla: Scalable Routing Protocols for Mobile Ad Hoc Networks. IEEE Network **16**(4) (2002) 11–21
8. M. Joa-Ng and I.-T. Lu: A Peer-to-Peer zone-based two-level link state routing for mobile Ad Hoc Networks. IEEE Journal on Selected Areas in Communication **17**(8) (1999) 1415–1425
9. 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
10. F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger: Geometric ad-hoc routing: Of theory and practice. In Proceedings of PODC (2003) 63–72
11. 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)
12. J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris: A Scalable Location Service for Geographic Ad Hoc Routing. Proceedings of the 6th Int. Conf. on Mobile Computing and Networking (MobiCom'00) (2000) 120–130
13. 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)
14. G. Pei, M. Gerla, X. Hong, C. C. Chiang: A Wireless Hierarchical Routing Protocol with Group Mobility. Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'99) (1999) 1538–1542
15. G. Pei, M. Gerla, and T.-W. Chen: Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks. Proceedings of IEEE International Conference on Communications **1** (2000) 70–74
16. C. Santivanez, S. Ramanathan, and I. Stavrakakis: Making Link State Routing Scale for Ad Hoc Networks. In Proceedings of MobiHOC'2001 (2001) 22–32
17. Y. Xu, J. Heidemann, and D. Estrin: Geography-informed Energy Conservation for Ad Hoc Routing. In Proceedings of Int. Conf. on Mobile Computing and Networking (MobiCom'2001) (2001) 70–84
18. B. Zhou, F. de Rango, M. Gerla, S. Marano: GeoLANMAR: Geo Assisted Landmark Routing for Scalable, Group Motion Wireless Ad Hoc Networks". In Proceeding of the IEEE 61st Semiannual Vehicular Technology Conference (VTC2005-Spring) **4** (2005) 2420–2424