

# GMR: Geographic Multicast Routing for Wireless Sensor Networks

Juan A. Sanchez  
Dept. Information and Comms. Eng.  
University of Murcia,  
jlaguna@dif.um.es

Pedro M. Ruiz  
Dept. Information and Comms. Eng.  
University of Murcia,  
pedrom@dif.um.es

Ivan Stojmenovic  
SITE, University of Ottawa  
Ottawa, Ontario, K1N 6N5  
ivan@site.uottawa.ca

**Abstract**—We present **Geographic Multicast Routing (GMR)**, a new multicast routing protocol for wireless sensor networks. GMR manages to preserve the good properties of previous geographic unicast routing schemes while being able to efficiently deliver multicast data messages to multiple destinations. It is a fully-localized algorithm (only needs information provided by neighbors) and it does not require any type of flooding throughout the network. Each node propagating a multicast data message needs to select a subset of its neighbors as relay nodes towards destinations. GMR optimizes cost over progress ratio. The cost is equal to the number of selected neighbors, while progress is the overall reduction of the remaining distances to destinations. That is, the difference between distance from current node to destinations and distance from selected nodes to destinations. Such neighbor selection achieves a good trade-off between the cost of the multicast tree and the effectiveness of the data distribution. Our cost-aware neighbor selection is based on a greedy set merging scheme achieving a  $O(Dn \min(D, n)^3)$  computation time, where  $n$  is the number of neighbors of current node and  $D$  is the number of destinations. This is superior to the exponential computational complexity of an existing solution (PBM) which tests all possible subsets of neighbours, and to an alternative solution that we considered, tests all the set partitions of destinations. Delivery to all destinations is guaranteed by applying face routing when no neighbor provides advance toward certain destinations. Our simulation results show that GMR outperforms previous multicast routing schemes in terms of cost of the trees and computation time over a variety of networking scenarios. In addition, GMR does not depend on the use of any parameter, while the closest competing protocol has one parameter and remains inferior for all values of that parameter.

## I. INTRODUCTION AND MOTIVATION

A sensor is a low-cost tiny device that is able to sense its environment. A wireless sensor network consists of a group of such sensors that are able to communicate among them using wireless interfaces. These radio interfaces are required to be very efficient in terms of power consumption, and their capacity is lower than other well-known radio technologies (e.g. 802.11 networks). In most sensor network scenarios, these devices acquire data from the environment, and send it to other nodes for further processing and analysis. When such destinations are not within the radio range of the source node, intermediate sensor nodes are used as relays. Routing protocols for wireless sensor networks are used to transmit messages from sources to destinations. They can be classified as unicast, broadcast or multicast. Unicast routing is used to send a message generated by a sensor node to a single destination or sink. Broadcasting is used to send a message from a sensor node to every other node in the network. Multicasting is used to deliver messages from a

single source to a set of destinations. Multicasting protocols try to minimize the consumption of network resources. For instance, sending one copy of the data to each destination using unicast is not considered multicast routing.

Possible applications of wireless sensor networks are endless, including habitat monitoring, wildfire detection, pollution monitoring, etc. There are many scenarios in which the use of multicast is of great interest. It is common in many of those, that sensors are required to send the same report to several sinks whose positions are known in advance. Also, one of the sinks may wish to multicast the same packet to other sinks with the help of sensors from the network. In such scenarios, it is vital to count on an efficient multicasting mechanism being able to alleviate the overall consumption of resources in the network.

Providing efficient multicast routing in wireless sensor networks poses special challenges compared to unicast data delivery. The scarce network resources which are common to sensor networks require multicast routing protocols to compute very efficient multicast distribution paths making use of a minimal amount of control information. Such efficiency can only be obtained by sending as few copies as possible of each datagram to reach all the destinations. However, those efficient paths are harder to compute than unicast paths. In fact the problem of computing a minimal bandwidth consumption multicast tree in wireless multihop networks was recently proven [12] to be NP-complete. This becomes specially challenging when overhead needs to be kept low due to the limited battery, storage capacity, bandwidth and processing power of sensor nodes.

Geographic unicast routing protocols have proven to be very effective to provide unicast routing in such resource-constrained scenarios. They work with local information, require a low computational cost, adapt very fast to changing network conditions and are able to route messages with a very low control overhead. To preserve those good properties, we build our multicast routing protocol upon the same geographic routing paradigm. However, we need to solve a number of technical aspects which are specific for the multicast operation. These include selecting neighbors so that the cost of the tree is reduced, performing face routing to reach multiple destinations, etc.

In particular, the main contribution of the paper is the new heuristic neighbor selection scheme, which requires a low computational cost and is able to compute very efficient multicast paths. In addition, the protocol does not require any type of network-wide flooding and it is solely based

on local geographic information obtained from neighboring nodes. By selecting neighbors based on our cost-aware metric, the proposed protocol is able to outperform previous schemes in terms of the cost of multicast packet delivery, and the computational cost of computing such trees. Thus, given that the focus is on computing trees by efficient neighbor selection, we assume throughout the paper that positions of destinations are known to multicast sources.

The remainder of the paper is organized as follows: section II presents an overview of the state of the art in the topic. Section III describes the GMR protocol. Our proposed greedy neighbour selection algorithm is explained in section IV. We evaluate in section V the performance of our solution using simulation. Finally, section VI provides some conclusions and discusses open issues.

## II. RELATED WORK

There have been a lot of multicast routing proposals for ad hoc networks; each of them based on different design decisions. Unfortunately they cannot effectively fulfill the unique requirements of wireless sensor networks. They are mostly designed to deal with highly mobile nodes, with higher processing and storage capacity, and a much limited amount of nodes. However, they provided the foundations for some of the proposals which came out later on for sensor networks.

In general, multicasting protocols proposed for mobile ad hoc networks are not designed to work in a localized way, and most of them do not use position information. They can be divided into tree-based and mesh-based protocols depending on the forwarding structures they employ. Mesh-based protocols expand a multicast tree with additional paths that can be used to forward multicast data packets when some of the links break. Examples of those protocols are ODMRP [2], CAMP [3] and PUMA [5]. These protocols have proven to be particularly suited for scenarios with high mobility rates. However, the maintenance of these structures through periodic broadcasts and the large amount of nodes which are required to forward multicast data messages make them impractical for sensor networks. Tree-based protocols such as MAODV [8], ADMR [7] and AMRIS [9] do typically use a lower number of relay nodes, but the tree needs to be reconstructed when links break due to mobility of the nodes. In addition, they also rely on periodic flooding, which is a costly operation for a sensor network. So, in both cases, their non-localized operation produces an excessive control overhead for a sensor network.

There have also been some proposals taking advantage of position information to perform multicast routing. the DDM [1] protocol works in an opportunistic way, which resembles very much the operation of geographic routing protocols. Destinations are included in the data packet and nodes decide next hops for each data packet based on the distance of the destination measured in number of hops from its unicast routing table. However, maintaining its unicast routing table requires the additional overhead of the unicast routing protocol itself. Other existing proposals such as LAM [4] (which is built upon TORA [6] and CBT [10]), make extensive use of broadcast, being impractical for wireless sensor networks.

Mizumoto et al. [13] proposed a protocol to send multicast messages to a geographic area rather than to multiple destinations. They use position information to build a multicast tree that aims at minimizing the number of links. However, as the wireless medium allows one to many communications, the cost of the tree is better characterized by the number of nodes than by the number of links. Thus, the resulting trees are not optimal.

Given the difficulty of adapting existing ad hoc multicast protocols to a fully-localized operation, new protocols have been recently proposed to meet those requirements. One of such protocols is Position Based Multicast Routing (PBM) protocol proposed by Mauve, Füßler, Widmer, and Lang, in [11]. This protocol, although not initially thought for sensor networks, fulfils most of the desired design criteria of localness and limited network overhead. It is a generalization of GFG (Greedy-Face-Greedy) [15] routing to operate over multiple destinations. It builds a multicast tree, whose shape can vary from the shortest path tree, to an approximation of a minimum cost multicast tree depending on a parameter denoted as  $\lambda$ . Authors in [11] try to find a good tradeoff between the total number of nodes forwarding the message and the optimality of individual paths towards the destinations. Each node evaluates all possible subsets of neighbors ( $W$ ) using a function to evaluate each  $w \in W$ .  $f(w) = \lambda N + (1 - \lambda)S$ ,  $0 \leq \lambda \leq 1$ , where  $N$  is the number of neighbors in the considered subset ( $|w|$ ) divided by the total number of neighbors ( $n$ ), and  $S$  is the summation of the minimal distances from nodes in  $W$  to destinations, normalized by the summation of distances from the current node to all destinations. From all possible subsets of neighbors ( $W$ ), the current node selects the one with optimal  $f(W)$ . If the best subset of neighbors is a single node, then that node will be the only relay for all the destinations. If a subset is selected, then each of the nodes in the subset will take care of routing the data messages to part of the destinations. If at some node there is no node providing advance towards one or more destinations, the authors use, only toward those destinations, a variant of face routing like the one we describe in section III-C.

The main problem with this approach is that determining the optimal value for  $\lambda$  is not a trivial task. In fact, the authors evaluated different values of  $\lambda$  but they never came out with a determination of an optimal value. An additional issue is the fact that the algorithm is computationally expensive. Evaluating all possible neighbor subsets has an exponential computational cost as the number of neighbors increases.

For networks with a very large number of multicast receivers, PBM may not scale well due to the need to include all destinations in multicast data packets. Scalable Position Based Multicast for Mobile Ad-Hoc Networks (SPBM [14]) was designed to improve scalability. It uses the geographic position of nodes to provide a scalable group membership scheme and to forward data packets. SPBM is mainly focused on the task of managing multicast groups in a scalable way. However, they fail to provide efficient multicast forwarding, because they use one separate unicast geographic routing for each destination. In addition, the interchange of routing tables between neighbors makes the protocol not so scalable to the number of multicast groups

as PBM is.

Given that the number of destinations is expected to be low for the multicast scenarios considered for sensor networks, our proposed scheme is mainly concerned with an enhanced neighbor selection mechanism. The proposed multicast protocol GMR selects neighbors based on the cost over progress framework that was first introduced by Kuruvila et al. [16] (for unicast routing) integrated with a greedy neighbor selection. Our cost function considers the number of transmissions based on the results of Ruiz et al. [12], that showed that the optimality of a multicast tree in terms of bandwidth consumption needs to be evaluated in terms of the minimization of the number of transmissions performed. Our scheme avoids hard-to-tune parameters, has lower computational costs, and computes multicast paths with a lower overall cost. Moreover, our scheme is general enough so as to be easily coupled with any scalable group management scheme.

### III. GEOGRAPHIC MULTICAST ROUTING

We present in this paper GMR, a new multicast routing protocol for wireless sensor networks. The problem we face can be described as follows. Given a multicast message generated by a source node, find a subset of nodes in the network so that the message is delivered to all destinations (sinks) with a very low consumption of resources. There are two kinds of resources to take care of, network bandwidth and sensor internal resources (i.e. battery, memory and CPU usage). We must minimize the number of messages sent, which means using a limited bandwidth and using as few sensors as possible to route the message to the destinations. It is important to design algorithms with a low computational cost, and a constrained memory consumption as the number of nodes increase.

To achieve those goals, our protocol will be based on the idea of geographic routing, which is able to meet those requirements. Of course, we adapt it so that it can deal with multiple destinations. As in previous geographic routing schemes, we assume that sensor nodes know their position, and they communicate their position to neighbors using periodic beacons. If the real position (e.g. GPS coordinates) is not available, virtual coordinates can also be used effectively [17]. The source is also assumed to know the position of the destinations as in previous geographic routing protocols.

Sensor nodes running GMR use the position of their neighbors to select which subset of them is best to propagate the message towards destinations. In our case the selected subset is the one that reduces most the total distance to destinations per unit of cost. When several neighbors are selected, each of them takes care of routing towards part of the overall set of destinations. When for some destinations no neighbor of the current node can reduce the distance, the algorithm uses face routing to exit the local minima until a new node providing advance is found. Within the remainder of this section we explain the general operation of the protocol in detail. Section IV describes in detail the way in which forwarding neighbors are selected.

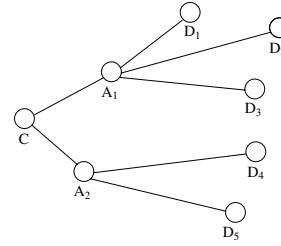


Fig. 1. Evaluating the candidate forwarding from C to A1 and A2

#### A. Network Model and Problem Formulation

In this paper we model a wireless sensor network as an undirected graph  $G = (V, E)$  where  $V$  is the set of vertices and  $E$  is the set of edges. The model assumes that the network is two dimensional (every node  $v \in V$  is embedded in the plane) and wireless nodes are represented by vertices of the graph. Each node  $v \in V$  has a transmission range  $r$ , which is equal for all nodes. Let  $dist(v_1, v_2)$  be the distance between two vertices  $v_1, v_2 \in V$ . An edge between two nodes  $v_1, v_2 \in V$  exists  $\iff dist(v_1, v_2) \leq r$  (i.e.  $v_1$  and  $v_2$  are able to communicate directly).

This model, known as unit disk graph (UDG), is a good initial approximation to make the problem tractable. Once a protocol is proven to be valid for the UDG, it can be fine-tuned to work with more realistic radio propagation models. As in previous work in unicast geographic routing, we assume that multicast sources know the position of all the destinations by some previous group management. That is, we focus on the path selection.

#### B. Cost Over Progress Metric

We now explain how the cost over progress metric can be used to select the next hops towards destinations. For clarity we first introduce it for the unicast case, and we will later explain how we extend it to multicast scenario.

In a unicast scenario, node  $C$ , currently holding the packet, will forward it to its neighbor  $A$ , closer to the destination  $D$  than itself, that minimizes the ratio of cost over progress. The cost function depends on the assumptions and metrics used, while progress measures the advance towards the destination.

Considering the multicasting problem, where a source node wishes to send a packet to a number of destinations (sinks) with known positions. It is assumed that the number of such destinations is small, which is a reasonable assumption for the considered scenarios. Unlike PBM, we describe here a solution which does not need any parameter. Assume that a node  $C$ , after receiving a multicast message, is responsible for destinations  $D_1, D_2 \dots D_5$ , and that it evaluates neighbors  $A_1, A_2$  as possible candidates for forwarding. The whole task can be sent to a single neighbour (e.g. if there exist one that is closer to all destinations than  $C$ ), or can be split to several neighbors, each with a subset of destinations to handle. Hop count is assumed to be proportional to distances.

Consider the case in Fig. 1 as illustration of the general principle. The current total distance for multicasting is  $T_1 = |CD_1| + |CD_2| + |CD_3| + |CD_4| + |CD_5|$ . If

$C$  considers  $A_1$  and  $A_2$  as forwarding nodes, covering  $D_1, D_2, D_3$ , and  $D_4, D_5$ , respectively, the new total distance is  $T_2 = |A_1D_1| + |A_1D_2| + |A_1D_3| + |A_2D_4| + |A_2D_5|$ , and the progress made is  $T_1 - T_2$ . Our aim is also to minimize the consumption of bandwidth, which is proportional to the total number of forwarding nodes selected. Thus, the cost is the number of selected neighbors, which in the above example is 2. Thus the forwarding set  $\{A_1, A_2\}$  is evaluated as  $\frac{2}{T_1 - T_2}$ . Among all candidate forwarding sets, the one with minimal value of this expression is selected. If there is no neighbor closer than  $C$  towards one or more of the destinations, then we have to enter into face mode. Section III-C describes how to proceed in this case.

### C. Multicast GFG

Multicasting is normally expected to proceed in greedy mode, it is, a node selects neighbours closer to the destinations than itself. However, a node  $C$  currently routing a message might not have any neighbour providing advance towards some of the destinations included in the message header. This situation is known as a local optimum for the greedy mode. Those destinations are included in a list called *multicast face-list*. In unicast geographic routing, a recovery scheme called face routing [15] is described. That algorithm finds an alternative route to escape from the local optimum until the message reaches a node where greedy mode can be resumed. The route toward that particular destination is said to be in perimeter mode during the search for a node closer to the destination than the node that experienced local minima.

When a node  $C$  has to route in perimeter mode for some of the destinations included in the message, it decides the next hop for every destination in the face-list according to GFG routing [15]. The protocol first decides which of the edges incident to  $C$  belong to a planar subgraph. One of such subgraphs normally used is the Gabriel Graph (GG), which contains edges  $CA$  such that the disk with diameter  $CA$  contains no other nodes inside it. Notice that node  $C$  may decide which of its edges belong to GG based only on the position of itself and its neighbors. It is not necessary to send any message for the purpose of constructing the GG. Face routing is applied independently toward each destination in the face-list. However, it may happen that the next selected neighbor for several destinations is the same. In this case the same relay takes care of those destinations in perimeter mode.

When a message with some destinations being routed in perimeter mode arrives at a node  $C$ , it checks whether its position is closer to any of the destinations than the node where the multicast perimeter mode started. If the test is positive, such destination is removed from the multicast face-list and added to the list of destinations to be routed in greedy mode. Multicast perimeter mode ends when the multicast face-list is empty.

It is also possible that the current node uses the same neighbor to forward traffic to different destinations being routed in different modes. That is, greedy routing may be in progress for some destinations, while perimeter mode can be in progress for others. Current node  $C$  in fact transmits a single message (counted as one in the overall cost), listing

all destinations, the mode being applied to each of them, and the neighbors that need to handle each of these destinations. As a result, a particular neighbor may be assigned to handle some destinations in greedy mode and some destinations in the perimeter mode at the same time.

### D. Packet Format

When multicast data is being forwarded, only those neighbours selected by the current node have to process the message. We add a GMR header to data messages to allow neighbours to realize that they are selected as relays. It is also used to mark which destinations require to be routed in perimeter mode.

That header contains the position of the sender and a list of fields, with one field associated to each of the neighbors selected as relay. A message transmitted by a node is received by all its neighbours due to the broadcast nature of wireless channels (assuming the considered unit disk graph model without collisions). This means that in order to deliver a multicast message to multiple relay nodes, we only need to send a single message with the appropriate information included on it. This property is commonly known as the *wireless multicast advantage* and it can help us to further reduce the overhead of our protocol. Every field contains the following information:

- Neighbour ID. The unique identification of the selected node.
- Number of Destinations in Perimeter Mode. This field contains the number of destinations for which this particular *Neighbour ID* is responsible in perimeter mode.
- Perimeter Information. This field is included only if there are any destination towards it is necessary to route in that mode. It contains information about the sensor in which the perimeter mode started and information related to face routing.
- Number of Destinations in Greedy Mode. This field contains the number of destinations for which this particular *Neighbour ID* is responsible in greedy mode.
- Destination List for Greedy mode. This is a list of sensor positions, one for every destination.

## IV. GREEDY NEIGHBOR SELECTION

In this section we gave a detailed explanation about the neighbour selection function. That is, what is the concrete algorithm used by the current node to decide which subset of its neighbors will forward multicast data messages for which subset of destinations. We describe that part of the protocol, showing the benefit in terms of computational cost compared to previous works.

Given  $k$  destinations, a possible algorithm can consider all the  $S_k$  partitions of the set of destinations. For each subset in a given set partition, the node checks whether it is possible to find a neighbor that is closer to all destinations in that subset than the current node  $C$ . If this is not possible for a subset, then this partition is ignored. If this is possible for each subset in the given set partition, then we measure the cost/progress ratio. Finally, after all the evaluations are made, we choose the best one among all the measured ones. This solution is applicable for small number of destinations,

e.g. up to 5. For larger number it becomes exponential in  $k$ , and therefore a faster greedy solution like the one presented below is needed.

We start with the set of destinations  $\{D_1, D_2, \dots, D_k\}$  for which there is a neighbor of the current node providing advance. We first group together, into the same subset, those destinations for which the neighbor providing the most advance is the same. For instance, in Fig. 2, the initial set partition to consider would be  $\{\{D_1, D_2, D_3\}, \{D_4, D_5\}\}$ , where  $A_1$  serves  $D_1, D_2, D_3$  and  $A_2$  serves  $D_4$  and  $D_5$ . In general, the set partition could be  $\{M_1, M_2, \dots, M_m\}$  with each destination being in exactly one of these subsets. Each  $M_i$  has its own cost-progress ratio, and the whole set partition also has its own cost-progress ratio as we explained before. The cost-progress ratio for the subset  $\{D_1, D_2, D_3\}$  in Fig. 1 can be computed as follows: The current total distance for multicasting is  $T_1 = |CD_1| + |CD_2| + |CD_3|$ . The new total distance is  $T_2 = |A_1D_1| + |A_1D_2| + |A_1D_3|$ , and the progress made is  $T_1 - T_2$ . The cost is the number of selected neighbors, which in the above example is 1. Thus, the forwarding cost over progress ratio is evaluated as  $\frac{1}{T_1 - T_2}$ .

Let  $P_i$  be the progress in  $M_i$  (each  $P_i$  is  $T_1 - T_2$  from above explanation). The cost in each  $M_i$  is 1 since all destinations in a given subset are served by the same (one) neighbor. The overall cost-progress ratio for the set partition is then  $\frac{1}{\sum_{i=1}^m P_i}$ . We are looking for the set partition with optimal ratio.

The greedy set partition selection algorithm for multicasting presented in algorithm 1 works as follows. First, the initial partition set  $M = \{M_1, M_2, \dots, M_m\}$  is initialized as we explained before. The current cost-progress ratio is computed accordingly as  $\frac{1}{\sum_{i=1}^m P_i}$ . The selection process then proceeds in rounds. In each round, all pairs  $\{M_i, M_j\}$  are checked for possible improvement over previously best cost over progress ratio. Two partitions can be combined only if there are neighbors of the current node, providing advance towards all the destinations in both partitions  $M_i$  and  $M_j$ . However, their merging, if possible, may not result in a better cost over progress ratio. The pair that provides the best improvement is selected and merging is performed, creating the new set partition  $M$ . The process continues to the next round and starts over again with new  $M$ . If no pair provided any improvement then the process stops and the best set partition  $M$  is found.

We now explain in more detail how two partitions merge, and how is *reduction* calculated. In order to merge two subsets  $M_i$  and  $M_j$  as one single set  $M_{i,j}$ , the algorithm considers the set of neighbors that are closer to all destinations in  $M_i \cup M_j$  than current node  $C$ . If that set is empty then merging is not possible, and *reduction* = 0. Otherwise, among all such neighbors, we select the one which provides the best cost over progress ratio for this new subset  $M_{i,j}$  (that is, the one that maximizes the corresponding progress  $BestReduction = T_1 - T_2$ ).

This algorithm, instead of testing all possible subsets, only needs to test  $O(D^3)$  of them (in the worse case), being  $D$  the total number of destinations. As discussed in the next section, when the number of neighbors of the current node

---

**Algorithm 1** Greedy set partition selection algorithm

---

```

1:  $M = \{M_1, M_2, \dots, M_m\}$  so that  $M_i = \{D_j \mid \text{same neighbour provides most advance}\}$ 
2:  $CRatio = \frac{1}{\sum_{i=1}^m P_i}$ 
3: repeat
4:    $BestReduction = 0$ 
5:   for all pairs  $\{M_i, M_j\}$  do do
6:     Find cost over progress reduction by merging of  $\{M_i, M_j\} \in M$ 
7:     if reduction >  $BestReduction$  then
8:        $BestReduction = \text{reduction}$ 
9:        $BestMerge = \{M_i, M_j\}$ 
10:    end if
11:  end for
12:  if  $BestReduction > 0$  then
13:     $M = \{M_1, M_2, \dots, \{M_i, M_j\}, \dots, M_m\}$ 
14:     $CRatio = \frac{1}{\sum_{i=1}^{|M|} P_i}$ 
15:  end if
16: until  $BestReduction = 0$ 

```

---

is lower than the number of destinations, there is no need to test more than  $n^3$  subsets, being  $n$  the number of neighbors.

We merge two subsets of destinations because it reduces the cost of retransmitting from 2 to 1. The node providing advance toward all destinations in the merged set may not improve overall progress significantly, but reduces forwarding cost in half, and the overall cost over progress ratio improves. We will now elaborate on this in more detail. Let's assume that  $M = \{M_1, M_2, \dots, M_m\}$  is the initial set partition in which all destinations in every  $M_i$  are served by the same closest node ( $B_i$ ) among all neighbors of the current node  $A$ . Let  $N(A)$  be the set of  $A$ 's neighbors. Given two subsets  $M_i, M_j \in M$ , we will analyze the conditions under which some neighbor with a lower progress can still provide a better tradeoff.

Let  $m$  be the cost of the initial partition, and  $\sum_{k=1}^m P_k$  be the progress made with such election. If we merge  $M_i$  and  $M_j$  into a single set served by a single node  $B_{i,j} \in N(A)$ ,  $B_{i,j} \neq B_i, B_{i,j} \neq B_j$ , the cost of the new subset  $M_i \cup M_j$  is 1. The overall cost after merging is then  $m - 1$ . In addition, the new progress made after merging would be  $\sum_{k \neq i, k \neq j} P_k + P_{i,j}$ , where  $P_{i,j}$  is the progress made by  $B_{i,j}$  towards destinations in  $M_{i,j}$ .

For the new cost-progress to be better than before merging, the following inequality must be satisfied:

$$\frac{m}{\sum_{k=1}^m P_k} > \frac{m-1}{\sum_{k \neq i, k \neq j} P_k + P_{i,j}}$$

Thus, the new overall progress of candidate neighbors to merge subsets must satisfy:

$$\sum_{k \neq i, k \neq j} P_k + P_{i,j} > \frac{m-1}{m} \sum_{k=1}^m P_k$$

This means that all those nodes  $x \in N(A)$  whose progress towards destinations satisfy above inequality can provide a better cost over progress ratio than the initial selection, even though they do not provide the best progress for any of the

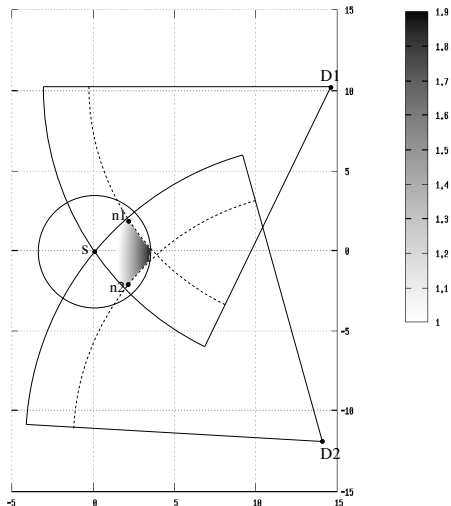


Fig. 2. Identification of nodes according to their goodness

destination sets. Every reduction of cost by merging two subsets allows to decrease the progress made up to  $\frac{\sum_{k=1}^m P_k}{m}$  units.

In figure 2 we can see current node  $S$  and two neighbors  $n_1$  and  $n_2$  within its radio range. We also see destinations  $D_1, D_2$ .  $n_1$  and  $n_2$  are the neighbors providing the best progress to  $D_1$  and  $D_2$  respectively. The shaded zone is the one in which there may exist other neighbors that can improve the overall cost over progress ratio. They do not provide the best advance for any of the individual destinations, but can provide a better trade-off. The legend of the graph represents the amount improvement in cost/progress ratio over the configuration before merging.

#### A. Algorithm Complexity

Low computation cost is very important in real sensor networks because of limited computational power available in currently existing devices. In this section we evaluate the worst case complexity of the neighbour selection algorithm and we show that it is asymptotically lower than that of PBM for most of the cases. Next section shows empirically that in the average case GMR also outperforms PBM. Also, the average complexity of the algorithm is much better than the worse case.

*Theorem 1:* The complexity of GMR is  $O(Dn \min(D, n)^3)$ , where  $D$  is the number of destinations and  $n$  the number of neighbors of the node currently multicasting the message.

*Proof:* The first step of the algorithm builds initial partition set  $M = \{M_1 \dots M_m\}$  by dividing destinations in subsets. Every subset  $M_i$  consists of those destinations whose closest neighbor of the current node is the same. That is, if a neighbor gives the best progress to two different destinations, those two destinations will be in the same initial subset. This stage needs  $D * n$  comparisons.

If the number of destinations is lower than the number of neighbours,  $D < n$ , then the number of subsets in the initial partition ( $m = |M|$ ) is, in the worst case, of size  $D$

(i.e. every subset is made by only a single destination). If the number of destinations is greater than the number of neighbours,  $D > n$ , then  $|M| \leq n$  because it is impossible to have more subsets than neighbours. In this case,  $|M_i| > 1$  for one or more  $M_i \in M$ . Thus,  $0 \leq m \leq \min(D, n)$ .

The process of computing the best subsets  $M_i, M_j \in M$  to merge requires testing each pair of subsets. The number of tested pairs is  $\frac{m*(m-1)}{2}$ . The largest number of iterations occurs when at every step it is always possible to find two subsets to be merged that improve the ratio. The total number of iterations in that case is  $m - 1$ . Then  $m$  may range from 1 to  $\min(D, n)$ . Given that iteration with  $m$  subsets may test  $O(\frac{m*(m-1)}{2})$  pairs, the worst case number of merging operations is  $\frac{m^3 - 6m^2 - m}{6}$ . This gives us an algorithm with  $O(m^3)$  merging steps. Merging two subsets may involve verifying all  $n$  neighbors for their feasibility, and testing  $O(D)$  destinations in two subsets being merged. Thus, in the worse case the algorithm has a complexity of  $O(Dn \min(D, n)^3)$ . ■

In the case of PBM, in all cases (i.e. worst, average, or best one) all possible subsets of neighbors are computed to select the best one. These neighbors are called forwarding nodes. For each destination, one of them takes responsibility for forwarding. Given  $n$  neighbors, the number of possible subsets is  $\sum_{k=0}^n \binom{n}{k} = 2^n$ . This leads to an exponential time complexity, which is much higher than the time complexity of selecting forwarding neighbors for GMR.

In addition, we must consider that in the average case of our algorithm, the cost is expected to be much lower, given that the worse case is highly improbable. The computation time is evaluated experimentally in the next section.

## V. PERFORMANCE EVALUATION

Our simulations compare our proposed cost over progress scheme, with six variants of the position-based multicast routing PBM [11]. These variants correspond to different values of the  $\lambda$  parameter. Being more specific, we use values of  $\lambda = 1, 0.8, 0.6, 0.4, 0.2, 0$ . The reason is that PBM behavior varies significantly depending on the  $\lambda$  selection, and there is no mechanism to find the best  $\lambda$  for a particular scenario. Authors of PBM showed that  $\lambda = 0.4$  is in their simulations the one apparently providing the best performance. However, as we shall see in this section, the best value of  $\lambda$  depends on specific network parameters. We consider a perfect MAC layer without collisions, and a unit disk graph model. That is, a message sent out by a node, is received by every other node in its radio range.

#### A. Simulation Setup

The simulation scenario consists of 1000 nodes randomly placed in an area of 1000x1000 m. Radio range was varied in order to achieve different network densities in terms of mean number of neighbours. Notice that this is equivalent to using a fixed radio range and increasing the simulation area. To vary density, we considered a mean number of 5, 6, 7, 8, 9, and 10 neighbours per node, whereas the number of multicast destinations was 5, 10, 25 and 50. Receivers were also randomly selected from the set of sensor nodes. For each configuration our results are the mean over a total

number of 50 simulation runs, which proved to be sufficient to provide a small 95% confidence interval.

### B. Performance metrics

To assess the performance of the proposed schemes we considered the following performance metrics:

- Number of Transmissions. This metric measures the efficiency of the multicast paths selected. The lower the number of transmissions, the lesser the network resources consumed to deliver the data message to all destinations. This is the most important performance parameter.
- Stretch Factor. The stretch factor is the maximum difference in hops between the shortest path tree to a receiver and the actual path. This metric provides an indication of how much does the protocol diverge from shortest paths. It can help at assessing how good the trade-off between cost and progress is.
- Number of times the protocol switches to face routing. This metric is useful to explain other performance results, as well as an indication on how good the neighbor selection is.
- Computation time. This metric evaluates how costly in terms of computational power is the protocol. It is also useful to compare the mean case performance of the different protocols in comparison with the theoretical worse case results we obtained in the previous section.

### C. Number of Transmissions

We analyze in this section the effectiveness of neighbor selection schemes. We shall see that our proposed scheme achieves a good trade-off between the lengths of individual paths, and the overall number of transmissions. That is, our scheme only splits paths to multiple destinations, when there is no cost-effective next hop towards the whole set of destinations. By using that cost-based path splitting strategy we are able to identify when it is really interesting to create different paths.

Fig. 3(a) shows that GMR outperforms PBM regardless of the values of the  $\lambda$  parameter and in a variety of network densities. The higher the density, the bigger the improvement obtained over PBM. As expected, the value of  $\lambda$  has an influence on PBM's performance. For values of  $\lambda$  close to 0 PBM tends to create shortest paths. But those paths are not optimal in terms of the number of transmissions. When  $\lambda$  approaches 1 PBM tries to minimize the number of transmissions regardless of the length of the paths. The problem with  $\lambda = 1$  is that by not considering progress to destinations, paths may become very large in terms of number of hops and number of transmissions. As we can see, even for those values of  $\lambda$  GMR manages to outperform PBM because GMR does not only minimize the number of transmissions but at the same time the mean path length. We can see in Fig. 3(a) that GMR achieves an improvement between a 2 to 35% for a mean density of 5 neighbors per node. The higher the density the biggest the improvement obtained by GMR. For instance, for a mean density of 10 neighbors per node, the improvement ranges between a 30 to 95%. In general, GMR is a 2 to 25% better than the best case of PBM (using  $\lambda = 0.2$ ) and a 38 to 95% better

than the worse case of PBM (using  $\lambda = 1$ ). The key to that improvement is the goodness of the neighbor selection function used by GMR. For higher densities, most of the routing is performed in greedy mode (almost no face routing is required). Thus, the better neighbor selection function achieves its higher advantage.

For different number of destinations, the results follow a similar trend. Of course, the higher the number of destinations the higher the number of transmissions required. But, in general the improvement achieved by GMR compared to PBM is in the same percentages.

Fig. 3(b) shows the percentage of destinations reached versus the number of transmissions for a mean density of 7. From it we can assess the efficiency in delivery for each protocol tested. The lower the number of transmissions needed to reach a certain number of destinations, the better the efficiency of the protocol. As we showed before, GMR clearly needs fewer messages to reach 100% of destinations than PBM using different  $\lambda$  values. This figure also shows that GMR makes a very efficient use of its transmissions because for a given number of transmissions, the percentage of destinations reached is higher. The reason for that is that GMR's neighbour selection algorithm manages to efficiently delay the splitting of paths to approximate the minimum number of transmissions tree whilst still providing a good advance toward destinations. Fig. 3(c) illustrates the different behaviours of GMR as density increases. As expected, the higher the density of the network the better the performance obtained. The reason is that with higher densities, the probability of entering into face mode drops rapidly. Hence, the neighbour selection algorithm becomes even more effective.

Summing it up, GMR behaves in any case better than the best case for PBM but, unlike in it, in GMR it is not necessary to adjust any parameter. GMR is a self adapting algorithm that, using only local information, is able to find out the right point in which to split multicast messages in order to maintain a good trade-off between path length and total number of messages transmitted.

### D. Stretch Factor

Usually achieving low number of data packet transmissions requires deviating from shortest path trees to allow many destinations to share the same paths as much as possible. For some applications with hard delay requirements, a big deviation from the shortest path tree may not be desirable. To account for the deviation from a shortest path tree, we compute the stretch factor for the resulting graphs from each of the protocols being evaluated.

For these experiments we use a mean density of 10 neighbors per node, to make the comparison as fair as possible. The reason is that for lower densities the protocols fall into face routing very often. So, the overall results for those cases will be mainly determined by the number of times the protocol entered into face routing rather than the neighbor selection itself. As we shall see our protocol outperforms PBM as GMR manages to fall into face mode a lower number of times than PBM.

Fig. 4(a) and Fig. 4(b) compare, in absolute and relative values respectively, the performance of the protocols for dif-

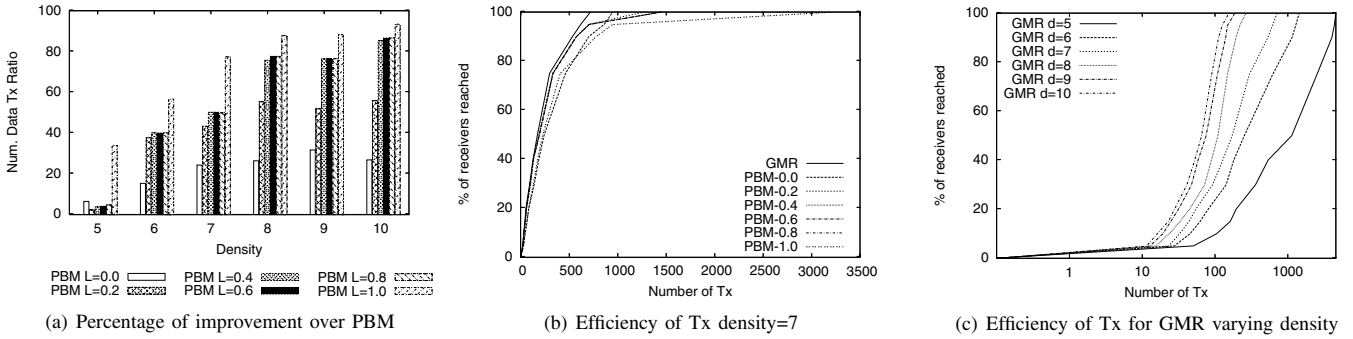


Fig. 3. Number of Transmissions for 25 destinations and varying density.

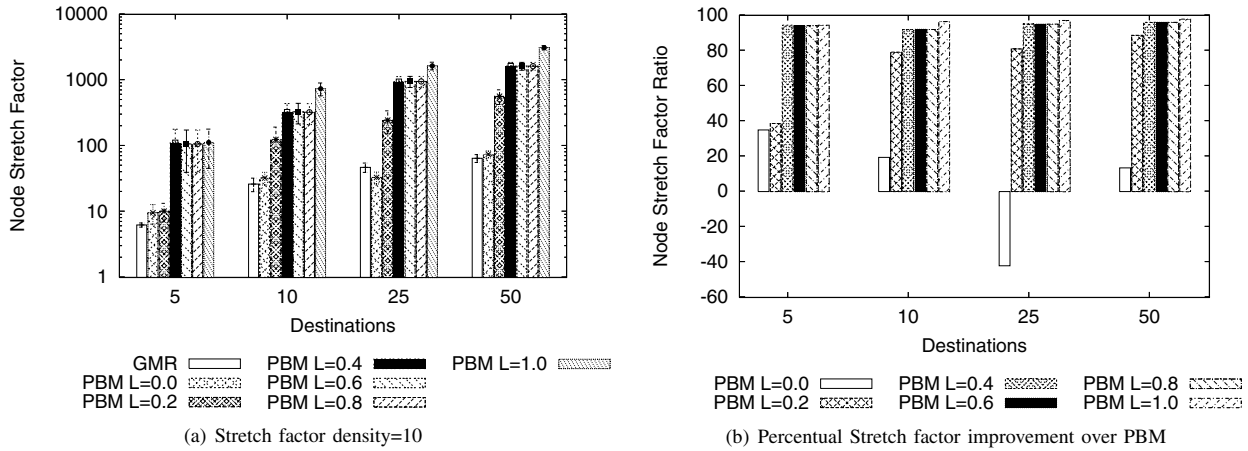


Fig. 4. Stretch factor comparison for a mean density of 10 neighbors per node

ferent values of  $\lambda$  and number of destinations. As expected, the lower the value of  $\lambda$ , the better the stretch factor PBM achieves. This is due to the fact that when  $\lambda$  approaches 0, PBM tries to compute a shortest path tree. In general, GMR is getting better stretch factor values than PBM for most of the tested configurations. Only in a few configurations in which  $\lambda = 0$ , GMR does not outperform PBM. The reason in those few cases is that the goal of GMR is not to find the shortest path but a good trade-off between the length of the paths and the overall consumption of network resources. As we can see in the figures, the trade-off obtained by GMR is very good because it is superior to PBM for every  $\lambda \geq 0.2$  whilst still needing a lower number of transmissions. In Fig. 4(b) we can see that the improvement in stretch factor achieved by GMR over PBM is bigger than a 40% for every  $\lambda > 0.2$ . For lower  $\lambda$  values, GMR still performs around a 15% better than PBM except in some particular cases like the one for 25 destinations shown in the figure.

#### E. Face Routing Fallbacks

As we mention before, when greedy forwarding falls into a local minimum we need to use perimeter mode to escape from it. Even though both PBM and GMR may use the same perimeter mode operation, the neighbour selection algorithm is completely different. These differences in the protocol can lead to different probabilities of reaching a void zone. Given that perimeter mode does not optimize any of the metrics,

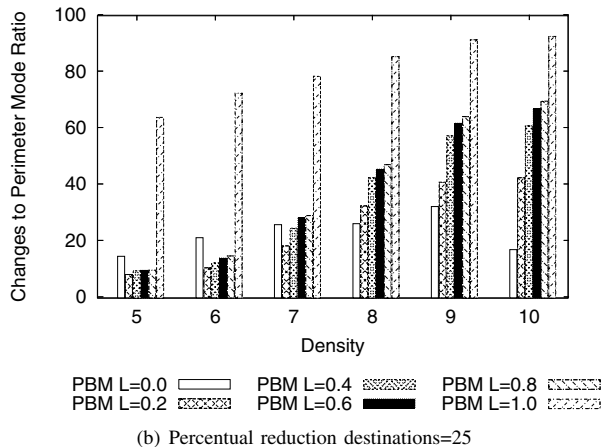
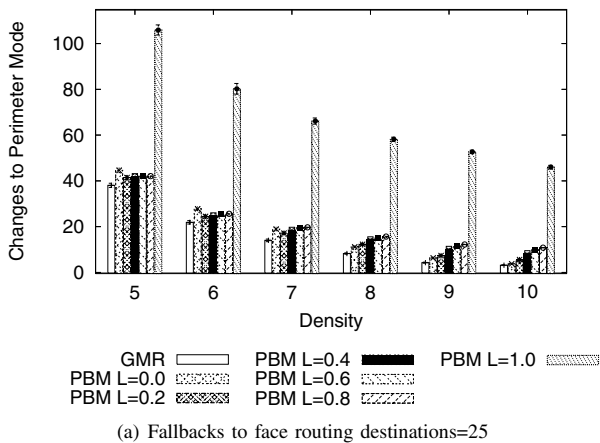
the fewer number of times the protocol enters perimeter mode, the better the overall performance achieved.

Fig. 5(a) shows the number of times each algorithm enters in perimeter mode. As we can see, for higher densities the number of times both algorithms enter in perimeter mode decreases. It is also noticeable that PBM makes more use of perimeter mode when  $\lambda$  values approach to 1 than for lower values of  $\lambda$ . Fig. 5(b) illustrates the improvement of GMR over PBM for the case of 25 destinations. We can see that GMR enters perimeter mode at least a 10% less times than PBM for any value of  $\lambda$ . The difference goes up to 80% in favour of GMR for the case of  $\lambda = 1$ . The higher the density the bigger is the improvement of GMR over PBM, although obviously both protocols enter into face routing a lower number of times.

#### F. Neighbor selection computation time.

As we showed before, the computational cost of PBM is exponential in  $n$  while the one of GMR is polynomial in  $n$  and  $D$ . Given that the average number of neighbors in our experiments is small (the highest density is 10), this may not translate into better performance for GMR. However, we show in our experimental results that the average case performance of GMR is very consistent and scalable across network parameters.

To measure the computational time, we execute one single step of the routing algorithm. This step makes the first



(a) Fallbacks to face routing destinations=25

(b) Percentual reduction destinations=25

Fig. 5. Changes to face routing for different densities and number of receivers

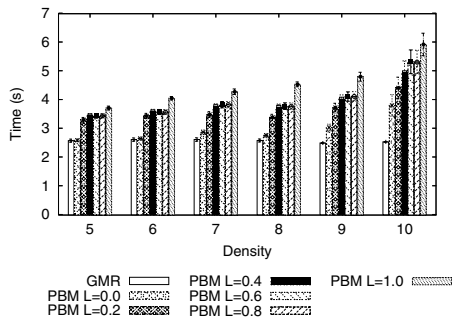


Fig. 6. Computation time for varying destinations.

neighbor selection (the initial one at the source node). For each set of parameters, we executed the single step 50 times and measured the total time of those 50 executions. We performed these 50 executions over 50 different graphs, being the computational time the mean on these 50 different graphs. Fig. 6 shows the results for the case that would be the best case for PBM.

We can see that the mean execution time for GMR is stabilized between 2 and 3 seconds regardless of the density of the network. However, PBM's mean execution time grows exponentially with density. This is because the proposed neighbor selection algorithm is able to compute a good set of neighbors efficiently, thanks to the subset merging heuristic. This shows that the mean case performance is much better than the worst case. On the other hand, the number of destinations does not significantly affect the computational time for neither of the algorithms.

## VI. CONCLUSIONS AND FUTURE WORK

We introduced GMR, a geographic multicast routing protocol for wireless sensor networks. GMR uses a cost-based neighbor selection at each routing step, allowing it to find a good trade-off between the optimality of the multicast tree, and the efficiency of data delivery. GMR is a fully-localized algorithm and does not require the use of extensive broadcast to route messages to a set of destinations. We assume the position of destinations is known by the source.

In fact, GMR has been designed for scenarios with a reduced number of receivers. Computing the optimal multicast tree in terms of bandwidth consumption is similar to finding the multicast tree with the minimum number of forwarding nodes, and was shown to be NP-complete (see [12]). Thus, the use of the greedy forwarding scheme is fully justified. In addition, using this greedy neighbor selection is also a good idea for sensor networks because the routing decision at each single node is performed based on the current network topology. Thus, GMR is able to adapt to topological changes due to sensors operating in a duty-cycle. However, individual sensors are assumed to be static. For future work, we will work on supporting mobility of the nodes themselves. One of the key aspects of GMR is the cost-aware neighbor selection function. The reason is that the shape of the tree is one of the key parameters governing the overall optimality of the tree. Given that the complexity of testing all possible subsets neighbor grows exponentially, we proposed an heuristic algorithm which has shown to be efficient compared to PBM. In addition, our performance results show that GMR outperforms PBM in terms of the efficiency of the resulting trees, the deviation from shortest path trees (stretch factor) and as expected on the computational time.

For future work, we will analyze different alternatives to perform face routing towards multiple destinations to assess which ones are more effective. In addition, we also plan to work on GMR variants (based on slightly different cost functions) to deal with energy efficiency and realistic physical layers.

## VII. ACKNOWLEDGEMENTS

Part of this work has been funded by Spanish MEC by means of the "Ramon y Cajal" workprogramme, the SAVIA (CIT-410000-2005-1) and SMART TIN2005-07705-C02-02 projects.

## REFERENCES

- [1] L. Ji, M.S. Corson, "Differential Destination Multicast: A MANET Multicast Routing Protocol of Small Groups," Proc. of IEEE INFOCOM'01, Anchorage, Alaska, April 2001, pp. 1192-1202.

- [2] S.J. Lee, W. Su, and M. Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 7, no. 6, December 2002, pp. 441–452.
- [3] J.J. Garcia-Luna-Aceves, and E.L. Madruga, "The Core-Assisted Mesh Protocol," *IEEE Journal on Selected Areas in Communications*, vol.17, no.8, August 1999, pp. 1380–1394.
- [4] L. Ji and S. Corson, "A Lightweight Adaptive Multicast Algorithm," *Proceedings of IEEE GLOBECOM'98*, Sydney, Australia, November 1998, pp. 1036–1042.
- [5] R. Vaishampayan, J.J. Garcia-Luna-Aceves, "Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks," in *Proceedings of MASS 2004*. in Proc. 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems MASS, Fort Lauderdale, Florida, USA, pp. 304–313.
- [6] V. Park and M.S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. In Internet Draft, draft-ietf-manet-tora-spec00. txt, Dec. 1997.
- [7] J.G. Jetcheva, D.B. Johnson, "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks," *Proceedings of ACM MobiHoc'01*, Long Beach, CA, October, 2001, pp. 33–44.
- [8] E.M. Belding-Royer and C.E. Perkins. "Multicast operation of the ad-hoc on-demand distance vector routing protocol," *Proceedings of ACM/IEEE MOBICOM'99*, Seattle, WA, August 1999, pp. 207–218.
- [9] C.W. Wu and Y.C. Tay: "AMRIS: A Multicast Protocol for Ad Hoc Wireless Networks", In Proc. of the IEEE Military Communications Conference (MILCOM-99), Atlantic City, NJ, November 1999. pp. 25–29.
- [10] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT) – An architecture for scalable inter-domain multicast routing," *Proc. of ACM SIGCOMM'93*, San Francisco, CA, October 1993, pp.85–95.
- [11] M. Mauve, H.Füßler, J. Widmer, T. Lang, "Position-Based Multicast Routing for Mobile Ad-Hoc Networks," TR-03-004, Department of Computer Science, University of Mannheim, March, 2003.
- [12] P.M. Ruiz and A.F. Gomez-Skarmeta, "Approximating Optimal Multicast Trees in Wireless Multihop Networks," in *proc. 10th IEEE Symposium on Computers and Communications, ISCC 2005*, La Manga, Spain, June 2005, pp. 686–691.
- [13] A. Mizumoto, H. Yamaguchi and K. Taniguchi "Cost-Conscious Geographic Multicast on MANET", In *proc. IEEE SECON 2004*, Santa Clara, CA, USA. pp. 44–53.
- [14] M. Transier, H. Fübler, J. Widmer, Martin Mauve and Wolfgang Effelsberg, "Scalable Position-Based Multicast for Mobile Ad-hoc Networks," *First International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim 2004)*, San Jose, CA, USA, 2004
- [15] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *ACM Wireless Networks*, Vol. 7, no. 6, November 2001, pp. 609–616.
- [16] J. Kuruvila, A. Nayak, I. Stojmenovic, "Hop Count Optimal Position Based Packet Routing Algorithms for Ad hoc Wireless Networks with a Realistic Physical Layer," *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems MASS*, Fort Lauderdale, Florida, USA, pp. 398–405
- [17] A. Caruso, A. Urpi, S. Chessa, and S. De, "GPS-Free Coordinate Assignment and Routing in Wireless Sensor Networks," in *Proc. IEEE Infocom 2005*, Vol. 1. Miami, USA, March 2005, pp.150-160