GMP: Distributed Geographic Multicast Routing in Wireless Sensor Networks

Shibo Wu, K. Selçuk Candan
Department of Computer Science, Arizona State University, Tempe, AZ, USA
E-mail: {shibo.wu, candan}@asu.edu

Abstract

In this paper, we propose a novel Geographic Multicast routing Protocol *(GMP) for wireless sensor networks*[1]. *The proposed protocol is fully distributed and stateless. Given a set of the destinations, the transmitting node first constructs a* virtual *Euclidean Steiner tree rooted at itself and including the destinations, using a novel and highly efficient* reduction ratio *heuristic (called rrSTR). Based on this locally computed tree and the information regarding the locations of its immediate neighbors, the transmitting node then splits the destinations into a set of groups and calculates a next hop for each of these groups. A copy of the packet and the locations of the corresponding group of destination nodes are directed towards the corresponding hop. The simulation results on NS2 show that the average per-destination hop count obtained using GMP is comparable to the existing PBM [21] algorithm and significantly less than obtained by using LGS [5]. Most significantly, GMP requires* 25% *less hops and energy than alternative algorithms.*

1. Introduction and Related Work

Geographic unicast routing protocols [4, 13, 31, 17] use location information to eliminate expensive wireless network operations. In contrast to single-source single-destination unicast schemes, group communication schemes like geocasting [15, 2, 28] and multicasting aim at identifying one-to-many transmission paths. Multicasting (as opposed to multiple unicasting) preserves network resources by reducing redundant messaging.

Most existing multicast routing protocols maintain a distributed structure for the delivery of multicast packets. In tree-based structures [7, 3, 30] there is only one path for each destination; multiple destinations may share parts of their paths. In mesh-based structures [18, 10, 19, 8], there may be multiple paths from a given source to each destination. Unfortunately, topology changes, node failures, and group membership changes can render the communication and reconfiguration overheads of maintaining a distributed tree or mesh structure unacceptably high.

[1]This work is supported by NSF grant # 0308268, "ARIA - Quality-Adaptive Media-Flow Architectures for Sensor Data Management"

In contrast, in source-routing based schemes (such as *Dynamic Source Multicast*, DSM [6]), the entire multicast tree is created by the source node in advance and included in the packet. In DSM, a *minimum spanning tree* based heuristic is used to create this routing graph. Each receiving node on this path decodes the multicast tree information and routes the packet to the next nodes as decided by the source. Unlike DSM, in *Location-guided Trees* (LGT) [5], each node only needs to know its own location and the locations of its neighbors. The source node locally constructs a multicast tree consisting of itself and the destination nodes. After receiving a packet, each subtree root extracts the corresponding set of destinations and then repeats the same process to partition this set into subsets of destinations. [5] presents two tree construction algorithms based on this underlying scheme: *location-guided k-ary tree (LGK)* and *location-guided Steiner tree (LGS)*. LGS approximates the Stenier trees using minimum spanning trees (MSTs) of the destination nodes. We note that the LGS scheme overly constrains the multicast trees it can generate. In particular, since each node creates potential multicast trees using only the destination nodes themselves, the trees that it can generate are limited. In contrast, in this paper, we propose a scheme which does not put the same constraints on the multicast trees explored. *Position Based Multicasting* (PBM) [21] is another protocol which makes forwarding decision based on local knowledge. Unlike LGS, however, it jointly optimizes (a) the progress of the packets towards the destinations and (b) the bandwidth usage. By considering all possible subsets of its neighbors and assigning each destination to the closest neighbor in the subset, PBM identifies a subset which minimizes the optimization criterion. Note that since each possible subset of the neighborhood has to be considered, the PBM algorithm can be very costly when there are large numbers of neighbors and destinations. Furthermore, tradeoff between remaining overall distance towards the destinations and bandwidth usage is not trivial.

1.1. Contributions of this Paper

In this paper, we propose a *virtual* Euclidean Steiner tree based multicast routing protocol where (a) transmitting nodes do not require any global knowledge to create a tree which will be used for partitioning the destination nodes

into groups and (b) they use only local information during actual route selection. Note that the general Euclidean Steiner tree problem is NP-hard [14]. However, a special case where there are only three nodes, the Steiner point can be calculated efficiently as in [24, 11]. Our algorithm, *Geographic Multicast routing Protocol* (GMP), exploits this to create heuristic Euclidean Steiner trees efficiently (in polynomial time). The progressive nature of the routing scheme enables continuous refinement of the resulting trees, thereby providing better multicast trees than similar schemes discussed above.

As discussed above, while creating the minimum spanning tree for partitioning the destinations, LGS [5] does not consider any geographic points other than the actual destinations themselves. PBM [21] tries to balance the per-destination hop count with the total number of hops needed to reach all destinations, with the help of a trade-off parameter. The optimal value of this parameter, however, changes from task to task and depends on the number of neighbors and the distribution of destinations. Thus, choosing a single suitable parameter value is not easy.

In this paper, we propose GMP, an efficient Euclidean Steiner tree based geographic multicast routing protocol. The underlying idea of GMP is that

> each transmitting node constructs a heuristic Euclidean Steiner tree, including the source and all destinations. The tree is *virtual* in the sense that it may include interior vertices that do not correspond to any actual wireless sensor nodes.

The destinations are divided into groups based on this tree. As in LGS a copy of the packet is then forwarded to a suitable next hop and the process is repeated by the receiving nodes until all destinations are reached. Essentially, there are two main differences between GMP and LGS: (1) GMP uses an efficient and effective heuristic to construct Euclidean Steiner tree (which allows all possible Euclidean points) while splitting destinations into partitions and (2) the subdestination (root of the subtree) towards which the packet is forwarded is constrained to be an actual destination in LGS, while in GMP it can be any Euclidean Steiner point. As shown in Section 5, these flexibilities result in significantly better multicast trees.

2. Wireless Network Model

In this paper, we adopt a commonly used sensor network model [31, 13, 32, 27]: A set, S, of nodes is located in a two dimensional geographic area, G. Each node $v_i \in S$ has coordinates, $coord(v_i) = \langle x_i, y_i \rangle$. Each node knows its own coordinates. This can be achieved either through an internal GPS device or through a separate calibration process [13]. The location of a node acts as its ID and its network address. Therefore, there is no need for a separate ID establishment protocol. Each packet is marked with the location of the next hop and the corresponding node picks up the packet. The source node (generally a prime node) knows the destinations prior to the dissemination of the data packet. In literature, there are works ranging from static group membership [12, 6] to highly dynamic scenarios supported by the source node [25, 5] or a separate group management service [20]. In this paper, we do not focus on the problem of how to establish and maintain multicast groups.

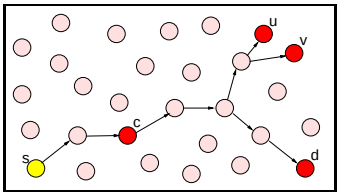


Figure 1. A multicast tree example. How the tree is created is described in Section 4.

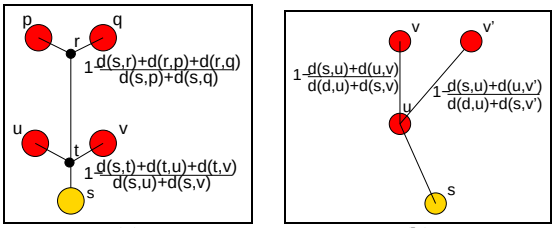
3. rrSTR: An Efficient *REDUCTION RATIO* Heuristic for Euclidean Steiner Trees

The GMP multicast routing algorithm we introduce in this paper is based on Euclidean Steiner trees. Note that generating optimal Euclidean trees is a costly operation [14] which needs to be avoided. There are many heuristics [23, 1, 33] and approximation algorithms [34, 26] to address this problem. Although they are much cheaper then optimal solutions, most approximation algorithms are still too costly to be deployed at sensor nodes. Some existing heuristics [22] apply to rectilinear Steiner problems and some are minimum spanning tree based algorithms [23, 26, 33].

These general purpose heuristics do not necessarily fit well for geographic multicasting. In fact, we note that (a) due to the lack of up-to-date global knowledge of the state of the wireless network, the Steiner tree points computed by any algorithm are not likely to be actual hops that will be used in the resulting route and (b) each receiving node in the network will have the opportunity to readjust the Steiner tree based on its own position. Therefore, instead of relying on expensive approximations or general purpose heuristics, in this paper, we focus on the following observations:

- Observation 1: when two destinations are far away from the source but are close to each other, they are likely to share subpaths. For example, in Figure 1, destinations $\{u, v\}$ are likely to share subpaths.
- Observation 2: when the angle of the line segments connecting the source node and the destinations are small, the nodes are likely to share subpaths. For example, in Figure 1, destinations $\{u, v, d\}$ are likely to share subpaths with destination c.

Since when there are only three nodes, the exact Steiner point can be calculated efficiently [24, 11], the observations



(a) (b)

Figure 2. Reduction ratio

enables us to develop a novel algorithm to create heuristic Euclidean Steiner trees efficiently (in polynomial time).

3.1. Reduction Ratio

Before we introduce the proposed heuristic to create Euclidean Steiner trees, we first introduce a novel measure, called *reduction ratio*, which uniformly captures these two observations to guide the construction of Euclidean Steiner trees. Given a destination pair (u, v) and a source node s, reduction ratio, $RR(s, u, v)$, is defined as follows:

$$RR(s, u, v) = 1 - \frac{d(s, t) + d(t, u) + d(t, v)}{d(s, u) + d(s, v)}$$

Here t is the exact Euclidean Steiner point of these three nodes $\{s, u, v\}$ [24, 11]. In the rest of the paper, the source node s is implicit when we refer to reduction ratio. The reduction ratio measure has the following properties (the proofs are omitted due to space constraints):

- The value of reduction ratio is always less than 1/2.
- Given two equidistant destinations, the reduction ratio is larger if these two destinations are further away from the source. For example, in Figure 2(a), the reduction ratio of (p, q) is larger than the reduction ratio of (u, v).
- Given a pair of destinations, the reduction ratio is larger if the angle between the two line segments connecting the source node and the two destinations is smaller. For example, in Figure 2(b), the reduction ratio of (u, v) is larger than the reduction ratio of (u, v').

3.2. Basic rrSTR Algorithm

In Figure 3, we present an iterative algorithm, rrSTR, to construct a Euclidean Steiner tree, based on these properties of the reduction ratio measure. In this subsection, we provide an overview of this algorithm.

Given a set of destination points, initially, source node s marks all its multicast destinations as *active*; that is, none of the destinations are covered yet. In each iteration, the algorithm identifies a destination pair, (u, v), with the largest reduction ratio. Given this pair, rrSTR creates a *virtual destination* w at the location of the Steiner point of nodes $\{s, u, v\}$. Note that when there are only three nodes, the Steiner point can be calculated efficiently [24, 11]. The

rrSTR(s, $destv$, rr): This function calculates the Steiner tree for nodes s and destinations in $destv$

- s is the current node;
- $destv$, the list of all destinations;
- rr, the radio range of s;
- T is the tree to be returned;

{

1. set all destinations and all pairs of destinations to be active;
2. calculate the reduction ratios and Steiner points for all pairs of destinations with s;
3. repeat
   - find an active pair (u, v) with largest reduction ratio and let t be the Steiner point of $\{s, u, v\}$
   - if no such pair can be found, break;
   - if u is the same node with v
     - add edge $\overline{su}$ to T, deactivate node u;
   - else if t is collocated with s
     - add edges $\overline{su}$ and $\overline{sv}$ to T, deactivate u and v;
   - else if t is collocated with u/v
     - add edge $\overline{uv}/\overline{vu}$ to T, deactivate v/u;
   - else if both $d(s, u)$ and $d(s, v)$ less than rr
     - deactivate the pair (u, v);
   - else if $d(s, u)/d(s, v)$ is less than rr
     - if $rr + d(t, u) + d(t, v)$ is larger than $d(s, u) + d(s, v)$, deactivate pair (u, v);
     - else add edge $\overline{uv}/\overline{vu}$ to T, and deactivate v/u;
   - else $d(s, t)$ is less than rr, and $rr + d(t, u) + d(t, v)$ is larger than $d(s, u) + d(s, v)$
     - add edges $\overline{su}$ and $\overline{sv}$ to T, deactivate u and v;
   - else /**create a new virtual destination**/
     - create a new virtual destination w at t, activate w, add edges $\overline{wu}$ and $\overline{wv}$ to T, calculate the reduction ratios and Steiner points for all pairs (w, i), where i is an active destination
4. return T;

}

Figure 3. Reduction-ratio-based heuristic for Euclidean Steiner tree generation

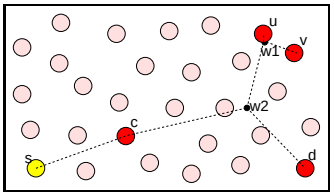
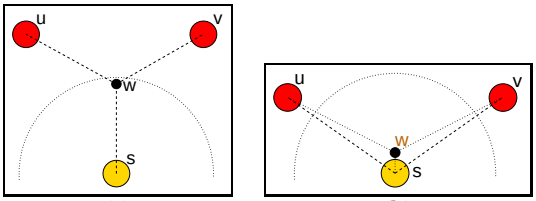


Figure 4. An example Euclidean Steiner tree generated by rrSTR

corresponding two edges, $\overline{wu}$ and $\overline{wv}$, are then added to the tree. Since they are already covered, both u and v are marked as *inactive* so that any destination pair that contains either u or v will not be considered in the remaining iterations. The new virtual destination w is added in the set of destinations and marked *active*. Thus, rrSTR will calculate reduction ratios for destination pairs consisting w and all remaining active destinations. Note that, in the extreme case, the Steiner point can be collocated with u or v. If for example the Steiner point is collocated with u, then no new virtual destination needs to be created. Instead, the edge $\overline{uv}$ is inserted into the tree and v is marked *inactive*. u stays *active*. Also note that the Steiner point can be collocated with the source node s itself. In this case, edges $\overline{su}$ and $\overline{sv}$ are inserted to the tree and nodes, u and v, are marked *inactive*.



(a) (b)

Figure 5. A virtual destination (a) may be or (b) may not be beneficial

Figure 4 illustrates the process with an example. In the first iteration, pair (u, v) is identified since they have the largest reduction ratio, and a virtual destination w_1 is created at the Steiner point of $\{s, u, v\}$. Edges $\overline{w_1u}$ and $\overline{w_1v}$ are added to the tree. Nodes u and v are then deactivated. In the second iteration, pair (w_1, d) is identified, and another virtual destination w_2 is created. Edges $\overline{w_2w_1}$ and $\overline{w_2d}$ are added to the tree. In the third iteration, pair (w_2, c) is identified. No virtual destination is created in this step since the Steiner point of $\{s, c, w_2\}$ is at node c itself. Instead, edge $\overline{cw_2}$ is added to the tree. At last, pair (c, c) is found and edge $\overline{sc}$ is added to the tree.

This algorithm is related to but different from conventional contraction based algorithms, such as [34, 26]. In this algorithms a full connected component is identified and replaced with a new point, iteratively. In rrSTR, instead, a destination pair is identified and replaced with a virtual destination. One difference is that the source node is never contracted. More importantly, the construction of the Steiner tree is guided with the *reduction ratio*, which identifies those pairs that are more likely to share subpaths.

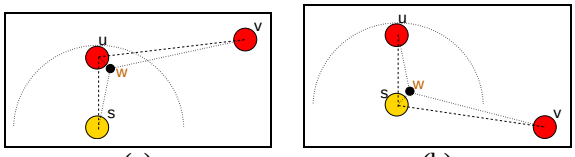
3.3. Radio Range aware rrSTR

The basic rrSTR algorithm described above uses the reduction ratio measure to guide the construction of Euclidean Steiner trees. However, the fact that it is not always good to use extremely short steps, especially within the *radio range* of a transmitting node, is not properly captured by this algorithm. Intuitively, when the Steiner point t is within the radio range, creating a virtual destination at t may not always be beneficial (Figure 5(b)). Therefore, in some cases the basic form of rrSTR may result in *redundant hops*. Thus, overly-eager virtual destination assignments should be avoided. The following are the three cases in which, given destinations u and v, it is not appropriate to create a new virtual destination:

- When both u and v are in the range of the current node, a new virtual destination would increase the number of hops to u and v by 1. Therefore, it is not appropriate to create a virtual destination at their Steiner point. Instead, we mark the pair (u, v) as *inactive*, so that this pair will not be considered in the future. Marking a pair (u, v) as *inactive* is different from marking two nodes u and v as *inactive*, in that pairs containing u or v other than the pair (u, v) can still be active.
- If neither u nor v, but the corresponding Steiner point t of $\{s, u, v\}$ is in the radio range of s, then a virtual destination may be beneficial in some cases. Since routing through the virtual destination will cost one hop on the resulting multicast tree, this will be acceptable only if

$$1 + \frac{d(t,u) + d(t,v)}{rr} < \frac{d(s,u) + d(s,v)}{rr},$$

  where rr is the radio range of current node. If the lefthand side is larger, then there is no benefit of using the virtual destination; therefore, we add edges $\overline{su}$ and $\overline{sv}$ and mark pair (u, v) as *inactive*. If the lefthand side is smaller, then we can create a virtual destination w at t. For example in Figure 5(a), it is beneficial to create the virtual destination w at t, but in Figure 5(b), it is not appropriate to create the virtual destination.
- When only u is in the radio range, using a new virtual destination at the Steiner point may not be useful. If the righthand side is larger, then instead w, u will be used as the Steiner point. In this case, edge $\overline{uv}$ will be added into the tree and v will be marked as *inactive*. If the left hand side is larger, s will be used as the Steiner point instead of w; edges $\overline{su}$ and $\overline{sv}$ will be added to the tree and both u and v will be marked *inactive*. For example, in Figure 6(a) the Steiner point becomes u, and in Figure 6(b) s becomes the Steiner point instead.



(a) (b)

Figure 6. It is not beneficial to create a virtual destination when only one of the destinations is in radio range

The algorithm presented in Figure 3 is radio range aware and implements these three special cases to prevent redundant hop generation and, thus, to save network resources.

4. GMP Routing based on rrSTR Trees

The outline of the GMP routing algorithm based on the rrSTR trees introduced in the previous section is presented in Figure 7. In this section, we describe how the GMP algorithm operates in detail. Let s be a source node.

Destination grouping: s first efficiently computes a *virtual* Euclidean Steiner tree as described in the previous section. s then uses this Steiner tree to split the destinations into

GMP($s, rr, pack$)
This function splits the destinations into appropriate groups and forward a copy of the packet to a next hop for each group of destinations

- s is the current node;
- rr, the radio range of s;
- $pack$ is the packet

{

1. extract destinations from $pack$ to $destv$;
2. set T to be rrSTR(s, $destv$, rr);
3. set $pivots$ to be the children of s in T
4. for each p in $pivots$;
   - find a neighbor n, that minimizes $d(n,p)$ subject to $\sum_{v(p)} d(n, v(p)) < \sum_{v(p)} d(s, v(p))$, where $v(p)$ is a non-virtual destination in the subtree rooted at p;
   - if n is found, clear PERIMODE flag in $pack$, remove p from $pivots$, forward a copy of $pack$ to n with all $v(p)$s saved in the copy;
   - else remove edge $\overline{pl}$ and add edge $\overline{sl}$, where l is the last child of p, add l to $pivots$;
     - if p has only one child o left, and p is a virtual destination, then remove edge $\overline{po}$ and add edge $\overline{so}$, add o to $pivots$, remove p from $pivots$;
     - else continue with the same p;
5. if $pivots$ is empty, return;
6. /** all destinations in $pivots$ are void now **/
   if PERIMODE flag is not set in $pack$, set the flag on;
7. calculate a next hop n by perimeter routing based on the average location of the destinations in $pivots$;
8. forward a copy of $pack$ to n with destinations in $pivots$ saved in the copy;

}

Figure 7. GMP routing algorithm

groups. We refer to the direct (terminal or non-terminal) descendants of s in the Steiner tree as *pivots*. Note that pivots may be actual nodes or they may be virtual, in the sense that a pivot may not always correspond to an actual sensor node. For each pivot p, s identifies all the non-virtual destinations in the subtree corresponding to this pivot. This set is referred to as the *group* of this pivot ($group(p)$).

Next hop selection: For each pivot p, s then identifies a next hop within its own neighborhood. In most cases, this node ($hop(p)$) is the neighbor closest to the pivot. However, to prevent routing loops, s also requires that the total distance from the next hop, $hop(p)$, to all destinations in $group(p)$ to be less than the total distance from s to all destinations in this group. For each pivot p, a copy of the packet as well as the destinations in $group(p)$ are sent to $hop(p)$. When the next hop receives the packet, it (1) extracts the corresponding destinations, (2) removes itself from this list if it is one of those destinations, and (3) repeats the above procedure, constructing a new Steiner tree to split the destinations into groups, and selecting next hops. This process is repeated by all hops until all the destinations are reached. Figure 8 illustrates the execution of the GMP algorithm:

1. In Figure 8(a), s constructs a Euclidean Steiner tree and chooses c as the pivot for destinations $\{c, u, v, d\}$. Then, s chooses n_1, the neighbor of s closest to c as next hop.
2. After a similar process, node n_1 forwards the message to node c, which in this case is both the pivot and the next hop (this step is omitted in the figure).
3. When node c receives the message, it extracts the destination list from the packet, and removes itself from the list. Based on the Steiner tree it constructs, as shown in Figure 8(b), it decides not to split the destinations and forwards the message to node n_2 which is closest to the pivot w_2 it computed for destinations $\{u, v, d\}$.
4. After a similar process, n_2 forwards the message to n_3 without altering the destination list.
5. n_3 constructs a virtual Euclidean Steiner tree (shown by the dashed line in Figure 8(c)). Note that, in this case, w_2, which is at the Steiner point of $\{n_3, w_1, d\}$, is not used as a virtual destination. This is because w_2 is in the radio range of n_3 and not close to the destinations $\{w_1, d\}$. Instead,
   - the pivot for destinations $\{u, v\}$ is w_1 and
   - the pivot for destination d is d itself.
6. A copy of the message is sent to the next hop n_4 for pivot w_1 with destinations specified as $\{u, v\}$. Another copy of the message is sent to n_5 with the only remaining destination d.
7. n_4 sends the message to u and v respectively.
8. n_5 sends the message to destination d.

The resulting tree was shown in Figure 1 in Section 3.

4.1. Dealing with Voids

Note that the process used to construct the Steiner tree does not consider the neighbor locations of the current node. Although this is not an issue in most cases (as an appropriate next hop can be found for each selected pivot in a dense network), it is possible that in some cases there will be no suitable neighbor with smaller total distance to the destinations in the pivot's group than the current node.

In GMP, the source considers if there exists a neighbor that has a smaller total distance to part of the destinations. If there exists such a neighbor, the source further splits the group into two smaller parts:

1. The source s removes the last child l of the pivot p from p's children list. (The last child of p can easily be found if the order in which edges are included to the Steiner tree is saved with along with this edge.)
2. s makes l a pivot by adding it to its own children list.
3. If there is only one child (say o) left in p's list of children, then
   - if p is a virtual destination, s makes o a new pivot and removes p from its children list
   - if p is an actual destination node, s does not remove p from its children list.

After splitting a group, next hops are calculated for each newly created or updated pivots. The splitting process continues if no valid next hop can be found for any pivot and its group. Figure 9 shows an example: Node s constructs the

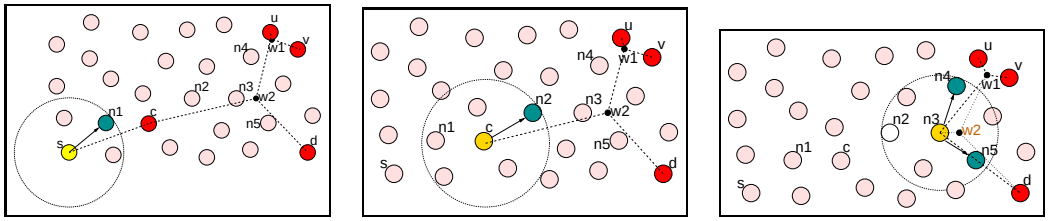


Figure 8. Example of GMP routing. The Steiner tree calculated by a current node is indicated by dashed edges.

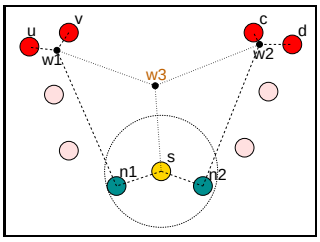


Figure 9. Splitting the set of destinations when there is no valid next hop

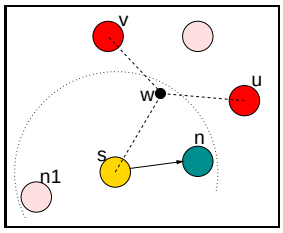


Figure 10. GMP allows *void* destinations to join groups with other nodes.

Steiner tree with w_3 as the pivot for the group of destinations $\{u, v, c, d\}$. However, neither one of the two possible neighbors, n_1 or n_2, has a smaller total distance to the destinations than s itself. Therefore, there is no valid next hop for this group. s will split the destinations into two groups and w_1 and w_2 will be assigned as new pivots. Since n_1 and n_2 are now valid next hops for the groups of w_1 and w_2 respectively, copies of messages will be sent to n_1 and n_2.

Consider a case where one or more groups contain a single non-virtual destination, where no neighbor is closer to any of the destinations than the current node. In unicast schemes (where by definition there is only one destination), a similar situation is dealt with by placing the packet into a *perimeter mode*[4, 13, 31]. Once in perimeter mode, the packet traverses the boundaries of the void area following the right hand rule until a node that is closer to the destination than the point where the packet enters the perimeter mode is reached. To correctly perform the right hand rule, the graph of the wireless nodes has to be planarized first, based on Relative Neighborhood or Gabriel Graphs [29, 9]. Such planarization can be done by the current node with only local information [4, 13, 31]. In GMP, since there may be multiple such destinations:

1. the source creates a single group which contains all such destinations and sets the packet for this group to be in perimeter mode.
2. s calculates the average of the geographic locations of these destinations and identifies the next hop in perimeter mode based on this average location, as in [21].
3. the packet is forwarded to this next hop with all destinations in this group recorded in the packet.
4. when a node receives a packet in perimeter mode, it first runs GMP to try splitting the destinations into groups and finds a valid next hop for each group.
5. If valid next hops are found for all groups, then the packet for each group is out of perimeter mode.
6. If no valid next hop can be found for any of the groups, the packet remains in perimeter mode and traverses the network with the same previous average destination.
7. If valid next hops are found for some but not all of the groups, then a new perimeter group will replace uncovered groups and a new average destination location is calculated for them. The packet starts a fresh round of perimeter routing with this new average destination.

The perimeter mode operation described above is similar to the one used by PBM [21] in the sense that the calculation of next hop in perimeter mode is based on the planarized graph of wireless nodes. However, GMP does allow a *void* destination, for which no suitable next hop exists, to join other destinations in a group, so that a valid next hop can be found for all these destinations as a group. In contrast, in PBM [21], once a void destination is identified, the packet for this destination will enter perimeter mode. We note that this is not always necessary. For example, in Figure 10, node s has no neighbor closer than itself to the destination v. In PBM, s will not group u and v together, and the packet for v will enter perimeter mode at s and is forwarded to n_1. In GMP, u and v are in one group and node n is a valid next hop for this group. Therefore the packet is for-

SIM.PARAMETER	VALUE
Simulator	ns-2.27
Network size	1000m X 1000m
Number of nodes	1000
Channel data rate	1Mbps
Mac protocol	Mac802.11
Transmission power	1.3W
Receiving power	0.9W
Message size	128B
Antenna	OmniAntenna
Radio Range	150m

Table 1. Simulation setup

warded to n for destinations u and v. Node n then forwards the packet to u, which will forward the packet to v.

4.2. Complexity of the GMP and Comparison with LGS and PBM

The computational complexity of GMP has two parts: (a) construction of the Steiner tree and (b) next hop selection:

- Let us assume there are n destinations and m neighbors for the current node, s. For constructing the Steiner tree, we use a priority queue and a 2-D array to maintain the status of destination pairs. In a given iteration, if there is no virtual destination created, the complexity of this step is $O(\log n)$ for removing a node from the priority queue. There can be at most $O(n^2)$ such iterations. If a virtual destination created, then the complexity of such a step is $O(n \log n)$ for inserting $O(n)$ pairs into the priority queue. There are at most $O(n)$ virtual destinations. Therefore, the complexity for the construction of the approximate Steiner tree is: $O(n^2)O(\log n) + O(n)O(n \log n) = O(n^2 \log n)$.
- In the next hop selection step, the complexity for calculating a next hop for each pivot is $O(m)$, and there are at most $O(n)$ pivots.

Hence, the complexity of each step of the GMP algorithm is $O(n^2 \log n + n \times m)$. In contrast, since PBM [21] considers all subsets of neighbors, it is exponential in m. Therefore, GMP is significantly more efficient than PBM, especially for dense networks. On the other hand, the complexity, $O(n^2 + n \times m)$, of LGS is slightly less than that of GMP. This is expected as LGS limits the tree creation to the geographic locations of the network nodes. As we demonstrate in the next section, this over-constrains the possible trees and results in less effective multicasting.

5. Simulation Results

In this section, we present the results of the experiments we carried to evaluate and compare the performance of the proposed GMP multicast protocol against existing multicast algorithms PBM [21] and LGS [5] as described in detail in Section 1. To see the impact of radio range awareness, we also experimented with GMP^{nr}, the version of GMP in which radio range aware decisions have been turned off. For

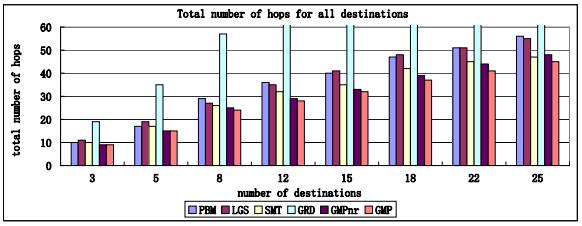


Figure 11. Total number of hops

the sake of completeness, we also implemented a centralized heuristic [16], denoted by SMT, to calculate the Steiner tree. This centralized algorithm assumes that the source node knows the positions of all sensor nodes in the network; thus the source node can calculate a close to optimal Steiner tree connecting itself and all destinations. The source node forwards a copy of the data packet with the routing information embedded in the packet. Naturally, acquiring the up-to-date global knowledge of the network topology is not practical for large sensor networks. Therefore, we include this centralized algorithm only for comparison purpose. Thirdly, we also experimented with GRD, which corresponds to the extreme case, where packets are independently routed for each destination. This algorithm explicitly minimizes the per-destination hop count and serves well as a lower-bound for the average number hops for each destination.

The simulation setup is described in Table 1. The NS2 simulator has been modified to support multicasting for wireless sensor networks. In each experiment, we generate 100 tasks. The 1000 nodes are uniformly distributed in the network. For each task, we randomly pick a node as the source node and randomly pick k nodes as the destination nodes. The value of k varied from 3 to 25. Each experiment is run on 10 different networks and results are averaged.

5.1. Total Number of Hops in the Multicast Tree

The total number of hops needed for a single multicasting task is the number of transmissions (forwarding) required to reach all destinations. Figure 11 shows the results obtained using four different multicast routing protocols: PBM, LGS, GMP and GMP^{nr}. In [21], it is noted that in PBM the minimum total number of hops is achieved for a λ value between 0 and 0.6. We have thus run the same routing task seven times, with the value of λ varying from 0 to 0.6. Among the results corresponding to these λ values, only the best (minimum number of hops) one is included for PBM in Figure 11.

In this figure, we see that, of the five protocols, GMP results in the least number of hops. Note that, GMP performs even better than the centralized algorithm, SMT. The reduction of GMP compared to both PBM and LGS is up to 25%. The results indicate that by using rrSTR destinations

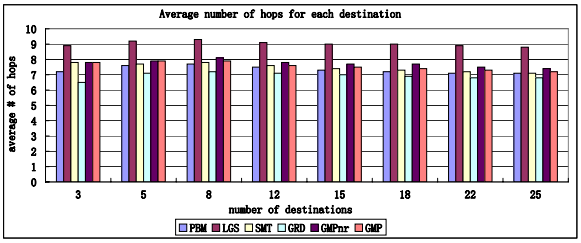


Figure 12. Per-destination hop count

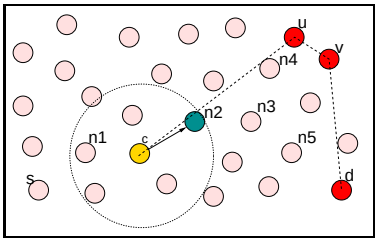


Figure 13. An example where LGS results in a large per-destination hop count

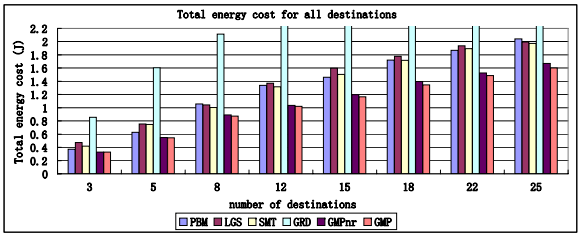


Figure 14. Total energy cost

are divided into groups more effectively and the way the next hop is calculated based on the Steiner point of the destinations in each group is more effective than the ways PBM and LGS calculate the next hops. As discussed in Section 3, when the radio range is not considered during the construction the Steiner tree, GMP^{nr} may generate redundant hops. The probability for redundancy becomes larger as the number of destinations becomes larger. Therefore, GMP^{nr} uses more hops than GMP. Note, however, that even without radio range awareness, GMP^{nr} works better than PBM and LGS.

As discussed earlier, in PBM the optimal value of the trade-off parameter λ depends on the number of neighbors and the distribution of the destinations. The value of λ, however, is fixed during routing, therefore, the destinations are split into groups by a receiving node sometimes earlier and sometimes later rather than being split by the node closest to the Steiner point of the destinations. Therefore PBM may generate a larger total number of hops. LGS uses a MST heuristic in which geographic locations other than locations of nodes in the network are not taken into consideration. Therefore the groups identified by LGS may not be as good as those identified by GMP. Furthermore, the calculation of next hop in LGS is based on one of the destinations in the group. In GMP, the next hop is calculated based on the Steiner point of the destinations in this group relative to the current node, therefore, this results in better paths.

5.2. Per-destination Hop Count

Figure 12 shows the average per-destination hop count. As discussed in Section 1, it is not always possible to minimize both the total number of hops and the per-destination hop count. PBM takes into account this by using a trade-off parameter, λ. When λ is small, PBM results in a smaller average number of hops and a larger total number of hops. When λ is larger, the average number of hops gets larger but the total number of hops becomes smaller. The value of λ in Figure 12 is the same as in Figure 11, i.e. the λ that minimizes the total number of hops.

We see that PBM, SMT and GMP provide comparable per destination hop counts (close to the greedy solution, GRD). LGS does not match the others in this respect. In LGS the next hop is chosen to be the neighbor closest to a destination that is closest to current node in the group. Routing in this way tends to reach the destinations sequentially and prevents the destinations from getting divided into groups at intermediate nodes. To see this, consider in Figure 13. Let us assume that the packet is at node c and the remaining destinations are $\{u, v, d\}$. The MST created by LGS will consist of the following three edges: $\{\overline{cu}, \overline{uv}, \overline{vd}\}$. Thus, node c does not split the destinations into groups and forwards the packet towards u since u is closest to c. Consequently, the path taken by the LGS will eventually be $c- > ...- > u- > v- > ...- > d$. Contrast this with the tree-structured multicast route computed by GMP as illustrated in Figure 1. Consequently, when LGS is used, destinations that are reached later may take a significantly larger number of hops and this increases the overall average number of hops.

5.3. Energy Consumption

Figure 14 shows the energy consumption during multicast routing.[2] We see that since the number of hops used by GMP is less than the number for all of PBM, LGS and SMT, the energy consumption of GMP is significantly smaller than consumptions of PBM, LGS and SMT in all configurations we tested. In these experiments, the energy savings compared to PBM and LGS is up to 25% as shown in Figure 14.

5.4. Failures Due to Perimeter Routing

When the network density is lower, the probability that there is no neighbor closer to the destination and that the packet enters the perimeter mode becomes higher. Because

[2]The energy consumption reported in this paper includes the transmission power of senders and the receiving power of all listening nodes within the transmission radio range of the senders.

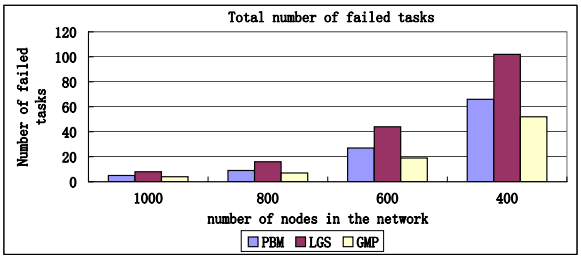


Figure 15. Number of failed tasks for different network densities

traversing the void area follows the right hand rule, the length of the path may become large. Furthermore, this may result in failed deliveries, especially when there is an upper bound associated with the total number of hops allowed for each packet. To observe how GMP scales against low density networks, we also did experiments to test the performance of GMP in networks with different network densities. In these experiments, the number of nodes in the network are 1000, 800, 600 and 400. The sensor nodes are uniformly distributed in the network. We set the maximum path length for each destination to be 100, that is a packet is dropped if its hop count reaches 100. Each task in these experiments routes a message from a random source to 12 randomly generated destinations. A task fails if not all destinations are reached. For each network size, we generated 10 networks and ran 100 tasks for each network. Figure 15 shows the total number of failed tasks in the total of 1000 tasks using three protocols PBM, LGS, GMP. (The other protocols do not use perimeter routing) We see that GMP has the least number of failures. LGS has the largest number of failures because it assumes a valid next hop can always be found and it fails when a void destination is identified even if the hop count is less than 100. PBM will group all the *void* destinations and always mark the packet to be in perimeter mode for these destinations. GMP, on the other hand, may group some *void* destinations with other destinations *without* setting the packet to perimeter mode for this group assuming a valid next hop can be found for this group as described in Section 4.

6. Conclusion

In this paper, we presented a novel *geographic multicast routing protocol*, GMP. GMP is built on an efficient and effective heuristic, rrSTR, for constructing radio-range aware Euclidean Steiner trees. Each transmitting node creates a virtual Steiner tree and uses it to choose subdestinations. These subdestinations are then used for identifying the next hops for actual packet forwarding. The computation complexity of GMP is significantly lower than that of PBM and comparable to that of LGS. On the other hand, simulation results on NS2 showed that the average number of hops on the multicast trees created by GMP is significantly lower than that of LGS and comparable to that of PBM. Therefore, GMP provides the best features of both alternatives. Furthermore, in terms of *total number of hops and energy consumption*, GMP is around 25% better than both of these alternatives.

References

[1] J. E. Beasley. A heuristic for euclidean and rectilinear steiner problems. *EJOR*, 58:284–292, 1992.

[2] J. Boleng, T. Camp *et al.* Mesh-based geocast routing protocols in an ad hoc network. *IPDPS*, pages 184–193, 2001.

[3] E. Bommaih, M. Liu *et al.* Amroute: Ad hoc multicast routing protocl. In *Internet-Draft*, 1998.

[4] P. Bose, P. Morin *et al.* Routing with guaranteed delivery in ad hoc wireless networks. In *DIAL-M*, 1999.

[5] K. Chen and K. Nahrstedt. Efficient location-guided tree construction algorithms for small group multicast in manet. In *INFOCOM*, pages 1180–1189, 2002.

[6] I. Chlamtac, S. Basagni *et al.* Location aware, dependable multicast for mobile ad hoc networks. *Computer Networks*, 36(5-6):2001, 659-670.

[7] M. S. Corson and S. G. Batsell. A reservation-based multicast (rmb) routing protocol for mobile networks. In *INFOCOM*, 1995.

[8] S. Das, B. Manoj *et al.* A dynamic core based multicast routing protocol for ad hoc wireless networks. In *MOBIHOC*, 2002.

[9] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[10] J. Garcia-Luna-Aceves and E. Madruga. "the core-assisted mesh protocol". *IEEE Journal on Selected Areas in Communications*, 17(8):1380–1394, 1999.

[11] R. Hwang, D. Richards *et al.* The steiner tree problem. *Annals of Discrete Mathematics*, 53, 1992.

[12] L. Ji and M. S. Corson. Differential destination multicast - a manet multicast routing protocol for small groups. In *INFOCOM*, 2001.

[13] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom*, pages 243–254, 2000.

[14] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[15] Y. Ko and N. H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *WMCSA*, pages 101–110, 1999.

[16] L. Kou, G. Markowsky *et al.* A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[17] F. Kuhn, R. Wattenhofer *et al.* Geometric ad-hoc routing: of theory and practice. In *PODC*, pages 63–72, 2003.

[18] S. Lee, M. Gerla *et al.* On demand multicast routing protocol. In *WCNC*, pages 1298–1302, 1999.

[19] S. Lee and C. Kim. Neighbor supproting ad hoc multicast routing protocol. In *MOBIHOC*, pages 37–50, 2000.

[20] J. Liu, J. Liu *et al.* Distributed group management in sensor networks: Algorithms and applications to localization and tracking. *Telecommunication Systems*, 26(2-4):235–251, 2004.

[21] M. Mauve, H. Füßler *et al.* Position-based multicast routing for mobile ad-hoc networks. In *MOBIHOC Poster*, 2003.

[22] I. I. Mändoiu, V. V. Vazirani *et al.* A new heuristic for rectilinear steiner trees. In *ICCAD*, pages 157–162, 1999.

[23] M. Minoux. Efficient greedy heuristics for steiner tree problems using reoptimization and supermodularity. *INFOR*, 28:221–233, 1990.

[24] J. Neuberg. Sur le point de steiner. In *Journal de mathématiques spéciales 1886*, page 29, 1886.

[25] S. Ratnasamy, B. Karp *et al.* Ght: A geographic hash table for data-centric storage in sensornets. In *WSNA*, 2002.

[26] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *Proc. Symposium on Discrete Algorithms*, 2000.

[27] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. In *ICC*, pages 1633–1639, 1998.

[28] I. Stojmenovic, A. P. Ruhil *et al.* Voronoi diagram and convex hull-based geocasting and routing in wireless networks. In *ISCC*, pages 51–56, 2001.

[29] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[30] C. Wu, Y. Tay C. Toh. "ad hoc multicast routing protocol utilizing increasing id-numbers functional specification". *Intenet-Draft*, 1998.

[31] S. Wu and K. S. Candan. Gper: Geographic power efficient routing in sensor networks. In *ICNP*, pages 161–172, 2004.

[32] Y. Xue and B. Li. A location-aided power-aware routing protocol in mobile ad hoc networks. In *IEEE Globecom*, pages 25–29, 2001.

[33] M. Zachariasen and P. Winter. Concatenation-based greedy heuristics for the euclidean steiner tree problem. *Algorithmica*, 25:418–437, 1999.

[34] A. Zelikovsky. Better approximation bounds for the network and euclidean steiner tree problems. In *Technical Report*, 1996.