Latency and Network Lifetime Trade-off in Geographic Multicast Routing for Multi-Sink Wireless Sensor Networks

Lucas Leão and Violeta Felea

FEMTO-ST institute, Univ. Bourgogne Franche-Comté, CNRS
DISC, 16 route de Gray, 25030 Besançon, France
{firstname.lastname}@femto-st.fr

Abstract. The deployment of multiple sinks in Wireless Sensor Networks may provide better reliability, timely communication and longevity, depending on the routing strategy. Moreover, geographic routing is a powerful strategy to avoid the costs related to maintaining high network knowledge. In this paper, we present a Geographic Multicast Routing solution (GeoM), focused on the latency and network lifetime trade-off. Our solution considers a linear combination of network metrics during the decision process of the next hop. Packets are forwarded to all sinks, and duplications are defined on the fly during the forwarding. The network lifetime is addressed with an energy balance strategy and a trade-off between progress and energy cost. We make use of the maximum energy consumption as an indication of the network lifetime. Simulation results show that GeoM has an overall better performance than the existing solutions, with improvements of approximately 11% for the average latency, and 54% for the maximum energy consumption.

Keywords: Multi-Sink Wireless Sensor Networks · Routing · Multicast · Latency · Network Lifetime · Geographic Routing.

1 Introduction

A Wireless Sensor Netowrks (WSN) is defined as a network composed of small wireless devices with limited capacity in terms of memory, processing and communication range. They share a common task of collecting and forwarding sensed data from various sources to a base station (sink) [2]. Applications of WSN may include environment monitoring in smart buildings and smart cities, military surveillance, planetary exploration etc.

The design of a WSN may consider different techniques for communication optimization, as for instance, the deployment of multiple base stations. A Multi-Sink WSN (MS-WSN) provides an immediate performance gain in terms of latency reduction [12]. Depending on the routing strategy, the deployment of multiple sinks also increases the network reliability, since packets may be forwarded to multiple different destinations.

In MS-WSNs, the final application drives the way communications take place. The information may be collected and forwarded following different models in order to cope with the application objective and requirements. As for instance, in order to meet a specific requirement such as reliability, the communication scheme may be designed as a many-to-many (any multiple sinks), or a many-to-all (all sinks) solution. These models require different routing specifications and strategies.

The routing perspective of the communication schemes are:

– **Unicast:** once the path towards a sink is defined, the route is reused until a predefined objective is no longer satisfied. As for instance, in real-time applications routes may change only when the maximum tolerable delay is surpassed.
– **Anycast:** information is addressed to a group of sinks. It can be 1-anycast, when information is forwarded to anyone of the sinks in the group, or k-anycast when information must reach any k sinks. The anycast routing may be applied to either increase network reliability (k sinks) or extend the network lifetime (any one sink).
– **Multicast:** information is addressed to all sinks in the network. The information of a single sensor node must be replicated and forwarded to each one of the sinks. Multicast approaches are focused on network reliability.

In this work we are interested in the case of MS-WSN with Multicast communication scheme, where packets must be forwarded to all sinks in the network. For that, we propose a Geographic Multicast Routing solution (GeoM), focused on reducing latency and increasing the network lifetime. Our algorithm differs from the literature by combining techniques to reduce latency and balance the energy consumption with multicast communication scheme and geographic routing. In geographic routing there is no need for routing tables or a setup phase, since routes are constructed on the fly based on the location information. In our solution, the next hop decision is taken using a linear combination of network metrics along with predefined weights. Based on the amount of consumed energy, the paths are constantly changed during the lifetime of the network, which balances the energy consumption. We also combine and adapt two different void handling techniques, making use of the right-hand rule and the passive participation [4].

The remainder of this paper is organized as follows. Section 2 presents a brief discussion on the existing works in MS-WSN routing. Section 3 describes our assumptions and the system model. Our solution is detailed in section 4, along with the testing scenarios and the discussion of the performance evaluation in section 5. We conclude the paper in section 6 with the future perspectives.

2 Related Work

There are few examples of works treating the many-to-all case. Most of the MS-WSN solutions are designed as unicast and 1-anycast routing protocols, dealing with the many-to-one and many-to-any cases. However, a unicast routing protocol could be used as a many-to-all solution if we assume that packets are duplicated at source and forwarded in sequence. The problem is that the energy consumption rises dramatically, since the number of packets forwarded individually is increased as a function of the number of sinks in the network. On the other hand, k-anycast solutions could be used as a multicast routing protocol if we assume that k equals the amount of sinks in the network. Because of that, in this section we describe both multicast and k-anycast solutions.

2.1 Multicast

In [5] the objective is to relay the messages from multiple sources to multiple sinks while reducing the number of links used to forward the messages. The strategy considers maximizing the overlap of paths from sources to sinks. This way, the same sub-routes, from the point of aggregation up to the splitting point, are used from different sources to forward messages to sinks. The next hop is defined by selecting the neighbors with more path overlaps and serving more sinks. The algorithm also considers a process of changing routes, which is an iterative mechanism that periodically triggers a search for new sub-routes.

The objective in [9] is to identify a routing tree connecting all sources to all destinations with the minimal number of links. The solution applies a searching method to identify an initial non optimal tree, which is improved by consecutive search iterations. The routes are initially constructed based on the hop distance to the sink. Each source has an independent path to each sink. Then, the algorithm tries to merge the independent paths from a source to all sinks, reusing as much as possible the same path. Later, the merge takes place with the sources, trying to determine a point of aggregation and replication. The convergence node is responsible to aggregate the message from the sources and re-split it to the sinks.

Although the works in [5] and [9] describe distributed solutions for multicast routing, they rely on heavy network knowledge in order to perform all the paths optimizations. They are also focused on data aggregation as a method to reduce energy consumption, and not explicitly oriented to latency minimization.

2.2 K-anycast

The authors in [3] propose RelBAS, a data gathering algorithm with a fault recovery scheme specially designed to assure reliability. The algorithm considers the construction of disjoint trees, rooted at each sink. However, the sensor nodes serving as forwarders are also disjoint, meaning that a node serves as forwarder to exactly one tree. In order to improve reliability, the solution considers forwarding the packet to exactly k sinks, which are decided in advance. The nodes are always part of exactly k trees, in order to forward the packet to the k sinks.

In [8] we find RPKAC, a routing protocol for Rechargeable MS-WSN designed to reduce network latency and optimize the energy consumption, but focused on assuring the delivery by forwarding the packets to k sinks. The authors establish strategies to forward the packets to at least k sinks, at most k sinks and exactly k sinks. The algorithm builds spanning trees rooted at the source node, with nodes sending route request messages in order to define the routing path. The neighbor nodes reply with their cost to reach a sink, and the current node decides to join an existing path with the lowest cost. The cost is calculated based on the linear combination of metrics: hop count, latency, energy cost and energy replenish rate.

The main objective in KanGuRou [14] is to guarantee the packet delivery to exactly k sinks and at the same time reduce the overall energy consumption. The strategy considers a geographic routing towards a set of sinks. The path is constructed in a greedy way. The current node builds a spanning tree to k sinks with minimum cost. The next hop is decided based on the cost of the energy-weighted shortest path (ESP). The solution assumes an adaptable transmission range in order to reduce the energy cost. At each hop, the algorithm calculates the cost over progress and decides to duplicate the packet towards different paths in order to reach k sinks. When a packet is duplicated, it is targeted to a set of specific sinks, in order to assure the delivery. The case of network void areas inherent to geographic routing is handled using a recovery mode with face routing, so packets can be forwarded out of the problematic area.

Routing solutions [3] and [8] are based on the hop-count distance, so the packets are forwarded to the neighbor with the lowest hop-count distance to the sink. This strategy implies either higher network topology knowledge or an important setup phase, with nodes discovering their hop-count distance to each sink. In [14] the focus is on reducing the energy consumption, favoring transmissions to closer nodes. Since the transmission range is variable, the energy cost to transmit a packet to a closer node is reduced. However, the amount of hops is increased and consequently the latency.

3 System Model and Assumptions

We represent a MS-WSN as a graph $G = (V, E)$, where V represents the set of all nodes and E the set of existing wireless links. Each $e \in E$ corresponds to a pair of nodes (i, j) as long as i and j, with $i \in V$ and $j \in V$, are within each other's transmission range (symmetric link). The set of neighbors of i is represented by V_i. We also specify that $V = S \cup N$ where S represents the set of sink nodes and N the set of sensor nodes, with $S \cap N = \emptyset$. The number of sinks is denoted by $|S|$.

We assume that every node is aware of its own geographic position and the geographic position of all sinks. For simplification, the geographic positions are represented by the Cartesian coordinates (x, y), and the distance is always the euclidean distance represented as $|ij|$ with $i, j \in V$. We also assume that packets are generated by sensor nodes only. For the energy consumption, we follow the radio model in [10], defining the consumed energy during transmission:

$$E_{T_x} = E_{elec} \times p + \epsilon_{amp} \times p \times d^2 \tag{1}$$

and the consumed energy during reception:

$$E_{R_x} = E_{elec} \times p \tag{2}$$

where E_{elec} is the dissipated energy for the transmitter/receiver electronics, ϵ_{amp} is the dissipated energy for the transmit amplifier, p is the amount of bits of the message and d is the transmission distance.

We consider two types of networks: with void areas and without void areas. A network with void areas is defined by the existence of a specific node out of the transmission range of a particular sink and for which there is no other neighbor node that presents a better geographic progress towards that sink than the node itself. In summary, $\exists\{n, s_i\}, n \in V, s_i \in S, (n, s_i) \notin E$ such that $|ns_i| < |v_j s_i|$ for all $v_j \in V_n$. Thus, the void area definition is for a node in relation to a sink. Because of that, in a MS-WSN it is possible for a node to be in the void area in relation to one or more sinks.

4 Geographic Multicast Routing

GeoM is a geographic routing protocol for Multi-Sink Wireless Sensor Networks that is capable of forwarding a generated packet to all available sinks in the network using a multicast communication scheme. It is focused on finding the trade-off between latency and the network lifetime. The next hop is selected by the current node based on the calculation of weighted metrics, and the highest intersection among forwarder candidates and sinks. The algorithm can be divided into two steps: filtering and selection. The first step is responsible for filtering the neighbor nodes in order to create a list of candidate forwarders. The filtering takes place by eliminating the neighbor nodes with negative progress and neighbors in void areas. If no neighbor is found, the recovery mode is triggered and a neighbor is selected using a void handling technique. At the same time, a broadcast message is sent to inform the neighbor nodes that the current node is in a void area. The second step is dedicated to effectively selecting the forwarders. The candidates are evaluated based on the weighted metrics, and a forwarder is selected based on the largest intersection of sinks. We also consider a metric called deviation factor, which increases the possibility of intersections by relaxing the constraints of the weighted metrics. Although GeoM does not require massive network knowledge, broadcast messages are used to periodically advertise the existence of neighbor nodes, their void area status and the amount of consumed energy.

4.1 Filtering Process

The filtering process starts with the reception of a packet. The first step is to check if the current node n is a target sink itself. In this case, the node must be removed from the list of destination sinks (S_r), since the packet has reached a

Algorithm 1 $GeoM(n, V_n, H, packet)$ - Run at the current node.

```
Input: $n$: current node, $packet$: packet to be forwarded, $V_n$: set of neighbors of $n$, $H$:
    set of neighbor nodes in void areas for a set of sinks
 1: $S_r \leftarrow get\_sinks(packet)$ /* Set of target sinks */
 2: $p \leftarrow get\_progress(packet)$ /* Progress of the previous hop */
 3: if $n \in S_r$ then
 4:     $S_r \leftarrow S_r \setminus \{n\}$
 5: end if
 6: if $S_r = \emptyset$ then
 7:     return
 8: end if
 9: $L \leftarrow \emptyset$ /* Set of pairs [candidate, sink] */
10: /* Check if the current node offers a positive progress towards the sinks */
11: /* A negative value means that the packet is in recovery mode */
12: $p_{new} \leftarrow ProgressTowards(n, S_r)$
13: if $p_{new} < p$ then
14:     /* Select a candidate with the recovery mode using right-hand rule */
15:     $L = Recovery(n, V_n, S')$ /* In this case $L$ has a single candidate */
16:     $ForwardersIntersec(n, L, S_r, packet)$
17:     return
18: end if
19: /* Look for the neighbors closer to the sinks than the current node */
20: $S' \leftarrow \emptyset$ /* Set of found sinks */
21: for all $v_j \in V_n$ do
22:     for all $s_i \in S_r$ do
23:         if $dist(n, s_i) > dist(v_j, s_i)$ and $\{[v_j, s_i]\} \notin H$ then
24:             $L \leftarrow L \cup \{[v_j, s_i]\}$
25:             if $S' \cap \{s_i\} = \emptyset$ then
26:                 $S' \leftarrow S' \cup \{s_i\}$
27:             end if
28:         end if
29:     end for
30: end for
31: if $0 < |S'| \leq |S_r|$ then
32:     $ForwardersIntersec(n, L, S', packet)$
33: end if
34: /* Check if a forwarder was found to all sinks */
35: $S'' \leftarrow S_r \setminus S'$
36: if $|S''| > 0$ then
37:     /* Notify neighbors about the existence of a void area */
38:     $SendVoidNotification(n, V_n, S'')$
39:     $L = Recovery(n, V_n, S'')$ /* In this case $L$ has a single candidate */
40:     $ForwardersIntersec(n, L, S'', packet)$
41: end if
42: return
```

target sink (algorithm 1, line 4). The list of target sinks must be composed of a set of unique and identifiable entities. This is important in order to assure that all packets are delivered to different sinks. If instead we had only a number of target sinks to reach, we would risk delivering all the packets to the same sink, because of the distributed aspect of our solution.

The second step is to check the progress of the current node in relation to the previous hop. If the current node (n) progress to the set of remaining sinks (S_r) represents a negative value compared to the value from the previous hop (line 12), it means that the packet was in Recovery Mode and no positive progress was yet found. In this case the packet must keep the Recovery Mode status and the forwarder candidate is selected using the right-hand rule [4]. The recovery strategy follows the same principles of the works in [11], [13] and [14]. In this case, only one node is selected and the packet is directly forwarded to this neighbor.

If the current node presents a positive progress towards the target sinks, the filtering step starts. Based on the neighbor list (V_n) of the current node, the algorithm creates a new list of candidate forwarders (L). The solution selects the neighbor nodes with a positive progress to at least one sink, and excludes the neighbors that have already announced being in a void area (line 23). The H list represents the set of pairs $[v_j, s_i]$ in which v_j is the neighbor node in a void area for the sink s_i. At the same time, a second sink list (S') is created in order to store the found sinks (line 26). If the number of found sinks is smaller than the amount of available sinks, it means that the algorithm could not find a suitable candidate to all sinks (line 35). In this case, the recovery mode is again activated in order to complete the list of candidates. A broadcast message (void announcement) is sent in order to inform the neighbors nodes that the current sink is in a void area (line 38). This strategy is important in order to allow future packets to avoid going through the void area.

4.2 Selection Process

The selection process is called when a forwarder candidate has to be selected. It is responsible for evaluating the list of candidates based on the weighted metrics and searching for the maximum intersection of candidates and sinks in order to avoid duplications. The first step is to calculate the weighted metrics to all candidates in relation to the sinks (algorithm 3, line 1). The aggregated decision metric W represents the value of the calculated weighted metrics to all candidates in relation to all available sinks.

The calculation process considers three network metrics and it is described in algorithm 2. The first is the relative distance from the candidate node and the target sink (line 13). The second metric is related to the energy cost for sending a packet from the current node to the candidate forwarder (line 14). The combination of these two metrics provides a balance between the progress towards the sink and the energy cost of that progress. Finally, the third metric is the total consumed energy of the candidate forwarder (line 15). This metric is used as a way to distribute the load over all the nodes and balance the energy consumption in order to avoid the early depletion of the battery. The total consumed energy of the neighbor is obtained from the received broadcast messages.

Algorithm 2 $W = CalculateMetric(n, S_r, L)$

```
Input: $n$: current node, $S_r$: list of remaining sinks, $L$: set of pairs [candidate, sink]
    with positive progress
Output: $W$: a matrix $[|S_r|, |V|]$ that stores the calculated weighted metrics
 1: /* $\alpha$: weight for the distance metric, $\beta$: weight for the energy cost metric, */
 2: /* $\delta$: weight for the consumed energy metric */
 3: $W \leftarrow \emptyset$
 4: for all $v_i \in \{v_i | [v_i, s] \in L\}$ do
 5:     for all $s_j \in S_r$ do
 6:         $D[v_i, s_j] \leftarrow dist(v_i, s_j)$
 7:     end for
 8:     $E[v_i] \leftarrow EnergyCost(n, v_i)$
 9:     $C[v_i] \leftarrow ConsumedEnergy(v_i)$
10: end for
11: for all $s_j \in S_r$ do
12:     for all $v_i \in \{v_i | [v_i, s] \in L\}$ do
13:         $x \leftarrow \alpha \times \frac{D[v_i,s_j] - Min(D[*,s_j])}{Max(D[*,s_j]) - Min(D[*,s_j])}$ /* for the distance between $v_i$ and $s_j$ */
14:         $y \leftarrow \beta \times \frac{E[v_i] - Min(E)}{Max(E) - Min(E)}$ /* for the energy cost from $n$ to $v_i$ */
15:         $z \leftarrow \delta \times \frac{C[v_i] - Min(C)}{Max(C) - Min(C)}$ /* for the energy consumption of neighbor $v_i$ */
16:         $W[s_j, v_i] = x + y + z$
17:     end for
18: end for
19: return $W$
```

The second step is related to the pre-selection of candidates based on their aggregated decision metric (algorithm 3, line 3). The algorithm fixes a sink and calculates the mean and the standard deviation to all candidates having that sink in their list. Later, a new list (C) is created having all the candidates that passed the pre-selection step for each sink. The pre-selection eliminates the candidates that have their aggregated decision metric with a value higher than the mean plus a factor (γ) of the standard deviation (line 9). By using a factor for the standard deviation the algorithm increases the forwarding angle, allowing non optimal nodes to be selected. This strategy is important to increase the possibility of intersections among candidates from one sink to another.

The third step of the selection process is responsible for calculating the largest intersection among the pre-selected candidates. This strategy tries to avoid early duplications, which translates to fewer packets being individually forwarded. In order to start the process, the closest sink (s) to the current node is selected (line 16) and all its candidates are inserted in the list ($P[p_k].nodes$). The sink s is removed from the list of remaining sinks (S_r) and included in the list of the already selected sinks (S_{used}). The algorithm searches for the next sink s by selecting the closest sink to an already selected sink (line 35). Later, the algorithm checks the list of candidates of the new s against the list of the already selected sink (line 27). If an intersection is detected, the sink s is added in the

Algorithm 3 $ForwardersIntersec(n, L, S_r, packet)$ - Run at the current node.

```
Input: $n$: current node, $L$: set of pairs [candidate, sink] with positive progress, $S_r$: list
    of remaining sinks, $packet$: packet to be forwarded
 1: $W \leftarrow CalculateMetric(n, S_r, L)$ /* The matrix that stores the weighted metrics
    */
 2: $C \leftarrow \emptyset$ /* The list that holds the structure having the candidate nodes */
 3: for all $s_j \in S_r$ do
 4:   $\mu_{W_{s_j}} \leftarrow mean(W[s_j, *])$
 5:   $\sigma_{W_{s_j}} \leftarrow stdev(W[s_j, *])$
 6:   for all $v_i \in \{v_i|[v_i, s] \in L\}$ do
 7:     /* Check if the calculated metric of the node $v_i$ for the sink $s_j$ is smaller */
 8:     /* than the mean of all other candidates plus a factor ($\gamma$) of the stdev */
 9:     if $W[s_j, v_i] \leq \mu_{W_{s_j}} + \gamma \times \sigma_{W_{s_j}}$ then
10:       $C[s_j].nodes \leftarrow C[s_j].nodes \cup \{v_i\}$
11:     end if
12:   end for
13: end for
14: $P \leftarrow \emptyset$ /* P is the list that holds the intersection structure */
15: $S_{used} \leftarrow \emptyset$
16: $s \leftarrow ClosestSink(\{n\}, S_r)$ /* Get the closest sink to $n$ */
17: repeat
18:   /* Go through P and search for intersection */
19:   for all $p_k \in P$ do
20:     if $P[p_k].nodes \cap C[s_j].nodes \neq \emptyset$ then
21:       $P[p_k].nodes \leftarrow P[p_k].nodes \cap C[s].nodes$
22:       $P[p_k].sinks \leftarrow P[p_k].sinks \cup \{s\}$
23:       $C[s].nodes \leftarrow \emptyset$
24:     end if
25:   end for
26:   /* If no intersection is found, a new entry is created (packet duplication) */
27:   if $C[s].nodes \neq \emptyset$ then
28:     $p \leftarrow |P| + 1$
29:     $P[p].nodes \leftarrow C[s].nodes$
30:     $P[p].sinks \leftarrow \{s\}$
31:     $C[s].nodes \leftarrow \emptyset$
32:   end if
33:   $S_{used} \leftarrow S_{used} \cup \{s\}$
34:   $S_r \leftarrow S_r \backslash \{s\}$
35:   $s \leftarrow ClosestSink(S_{used}, S)$ /* Get the closest sink to any selected sink */
36: until $S_r \neq \emptyset$
37: for all $p_k \in P$ do
38:   $v \leftarrow v_i \in P[p_k].nodes$ which minimizes $mean(W[P[p_k].sinks, v_i])$
39:   $SendPacket(P[p_k].sinks, v, packet)$
40: end for
41: return
```

P list corresponding to the intersection. Only the intersected candidates are kept in the $P[p_k].nodes$ list. However, if no intersection is detected, a new entry in the P list is created with the sink s and all its candidates. A new entry in P represents a duplication of the packet due to the impossibility of finding an intersection. The entire process is repeated iteratively for all sinks.

The final step considers that multiple candidates may still exist within an entry p_k of the P list, in the sense that a set of sinks $P[p_k].sinks$ may have more than one candidate forwarder $|P[p_k].nodes| > 1$. For that matter, the algorithm searches for the candidate forwarder that minimizes the mean considering the set of sinks $P[p_k].sinks$. This is the opposite view of the pre-selection step, where the mean was calculated on all candidates for a single sink. At this time, we calculate the mean on all sinks in $P[p_k].sinks$ for a given candidate $v_i \in P[p_k].nodes$. Once the candidate v_i is defined, the packet is sent.

5 Simulation and Results

GeoM protocol was developed using Contiki OS [6], which is a lightweight open source operating system designed for resource limited devices. Contiki OS is implemented in C programing language and provides portability to different platforms. We evaluated the performance of our solution through simulations using Cooja [6], which is a network simulator for the Contiki OS. Cooja is a Java-based application capable of emulating various sensor nodes [15].

The performance of our solution was compared to an existing approach (KanGuRou) [14], that was also adapted to Contiki OS and tested with Cooja under the same configurations. KanGuRou is a k-anycast solution, however it is capable of performing multicast routing, since k-anycast becomes multicast when k equals the amount of all sinks. The decision to use KanGuRou as a benchmark is related to the compatibility of assumptions. Both GeoM and KanGuRou are designed for geographic routing, which implies a small need of network knowledge, no routing table maintenance and no set up phase.

Table 1. Configuration for the simulations

Simulation Settings	
Deployment Density (d)	8 neighbors (on average)
Network Area (variable)	$\frac{\pi \times r^2}{d} \times \|V\|$ (3)
Communication Range (r)	50 m
# of Sensors ($\|N\|$)	50, 100, 150, 200, 250, 300
# of Sinks ($\|S\|$)	10% of the number of Sensors
# of Generated Networks	10 per scenario
# of Scenarios	6 scenarios with voids and 6 without voids
Packet Size	240 bytes
Packet Generation Rate	20% chance at every minute for each sensor
Radio Type	802.15.4
MAC Protocol	CX-MAC, modified version of [1]
Execution Time	120 minutes for each network

The simulation environment and details are outlined in table 1. In order to keep a similar deployment density over all variations of $|V|$, we make the network area vary with the number of deployed nodes, as given by equation 3 in table 1. We tested the solutions under different network topologies, varying the number of sensor nodes and the existence of void areas (10 simulations for each scenario).

The performance is evaluated by observing the average of all metrics for all simulations, and the results are presented with a confidence interval of 95%. The considered metrics include the Latency, defined by the average time a packet takes to be routed from the source to all sinks. It is calculated to each network considering the sum of all latencies divided by the number of received packets. We also analyze the Maximum Energy Consumption, which regards the node that consumed the largest amount of energy in the network at the end of the simulation. It gives an indication of the Network Lifetime, since it shows how far the node is from the entire depletion of its battery. As per explanation, we consider a network to be alive as long as all nodes have some energy. Therefore, network lifetime is considered to be the earliest moment at which a node's battery is completely depleted.

Two main network scenarios were considered: with void areas and without void areas. The results are presented the same way, and we vary the amount of sensors and sinks in order to evaluate the behavior of the solution when the network grows.

5.1 Networks without Void Areas

As we can notice in figure 1, GeoM presents a better performance in terms of

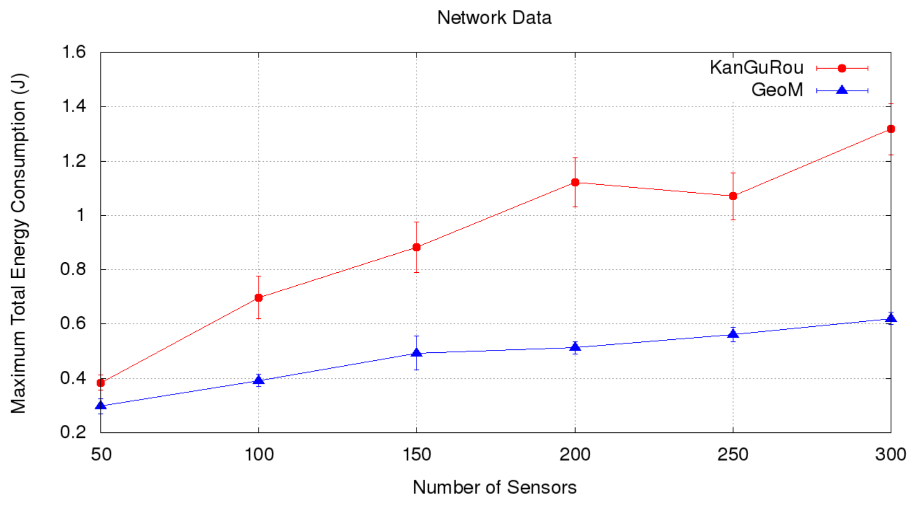

Fig. 1. Maximum Energy Consumption results with the network size varying from 50 to 300 nodes and sinks at 10% of the sensor nodes - networks without void areas.

Maximum Energy Consumption, with gains varying from 23% to 53%. For both solutions, the value for the maximum energy consumption increases when the network grows. However, the increase in GeoM is much smoother compared to KanGuRou. This is explained by our energy balance strategy and the deviation factor that distributes the load among other neighbors instead of using the same path for every forwarding. We can confirm this observation with figure 2, that shows the evolution of the maximum energy consumption during the simulation for the scenario with 300 sensor nodes. We can see that GeoM has a much smoother increase compared to KanGuRou. At the end of the simulation, GeoM presents half of the value for the maximum energy consumption. In terms of longevity, the maximum energy consumption for KanGuRou translates into a faster depletion of the node's battery.

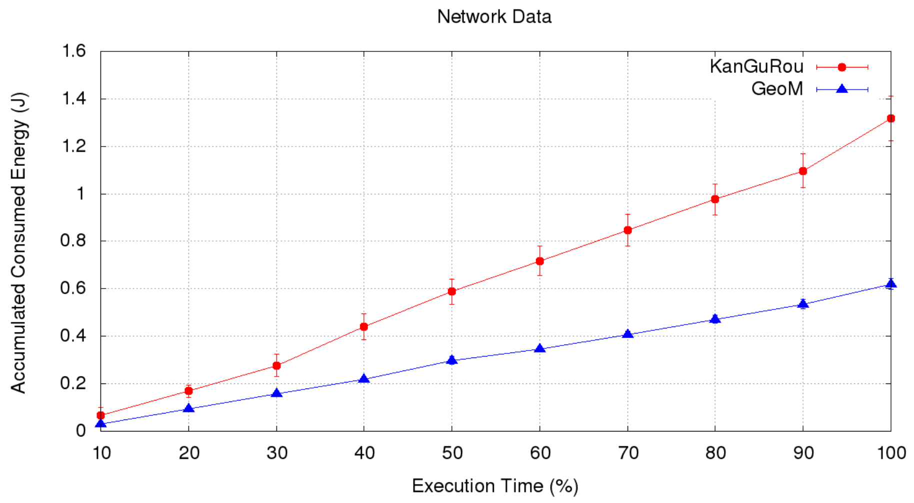

Fig. 2. Evolution of the maximum consumed energy. Results for 300 sensors and 30 sinks in a network without void areas.

In figure 3 we present the distribution of nodes in terms of energy consumption at the end of the simulation for the scenario with 300 sensor nodes. For KanGuRou, we can see a huge concentration of nodes with a small energy consumption, and a small group of nodes with a high level of energy consumption. For GeoM, we can see a different configuration, with a concentration of nodes at the first half of the range, which shows a better balance of the energy consumption.

The network may face many issues when a node's battery is completely depleted. The first one is the risk of disconnection, when part of the network is isolated and unable to forward packets to a sink. Even if a disconnection does not take place at a first moment, void areas may be created, since the depleted

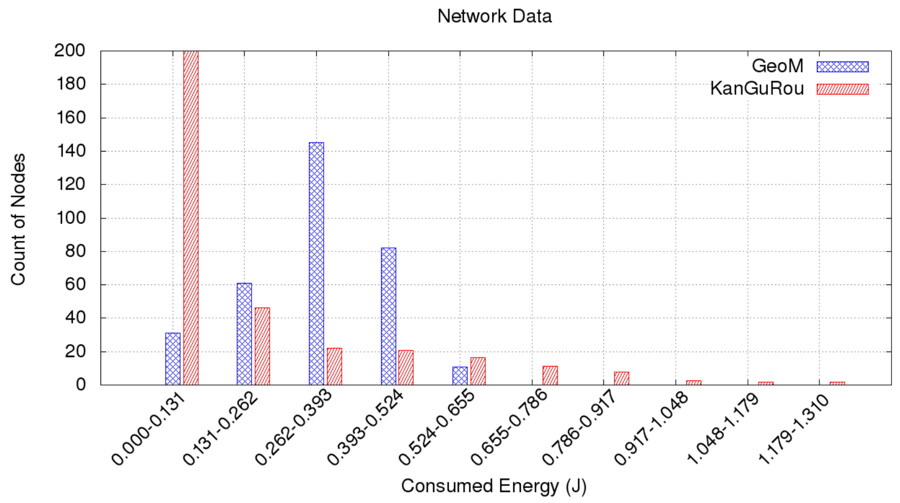

Fig. 3. Final State of the Network with the distribution of nodes in terms of consumed energy. Results for 300 sensors and 30 sinks in a network without void areas.

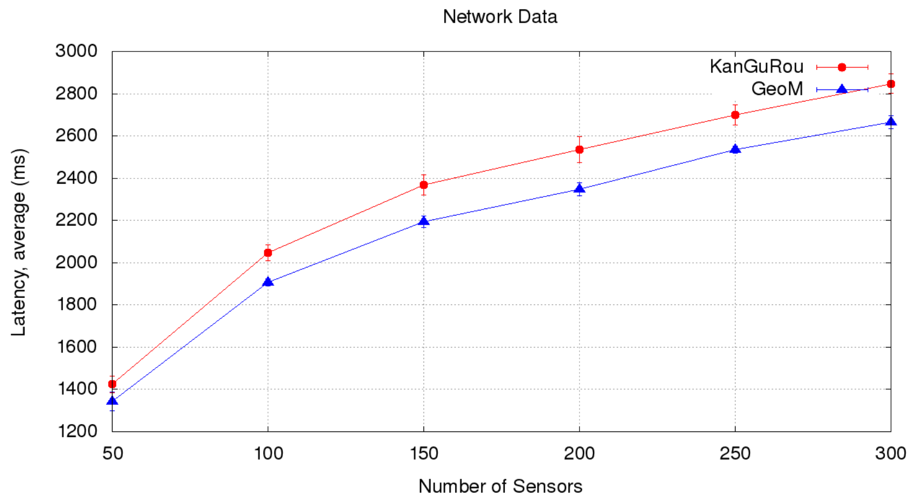

Fig. 4. Average Latency results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes in a network without void areas.

nodes are generally the ones within the sink neighborhood. The consequence of that is an increase of the latency and energy consumption, since a longer path is expected in order to bypass the void area. Another issue is related to the application, that faces a lack of coverage in the sensed field.

The Latency results are presented in figure 4. We can notice that even with the energy balance strategy, GeoM presents a performance gain of around 7%. This as a positive indication, since the energy balance and the deviation factor strategies normally increase the path length. The performance gain can be explained by the fact that GeoM duplicates the packets a little earlier than KanGuRou, which speeds up the delivery. This is part of the trade-off between better energy consumption and latency.

5.2 Networks with Void Areas

The existence of a void area is a problematic and inherent issue in geographic routing. Strategies must be applied in order to handle the packets entering in a void area. GeoM deals with void areas using two existing techniques, the right-hand rule and the passive participation (void announcements) [4]. When a packet is in recovery mode, after being forwarded to a void area, the next hop decision does not consider the energy and latency optimizations. Normally, the path to exit the void area is always the same, regardless the amount of time it is used. In order to avoid this scenario, GeoM makes use of the void announcements as a way to prevent packets from entering in a void area, whenever it is possible. The simulation results show that GeoM is able to handle void areas and provide performance gains for both the energy and latency.

In figure 5 we can notice a performance gain varying from 5% to 26%. Even with the existence of void areas, GeoM is capable of distributing the loads and

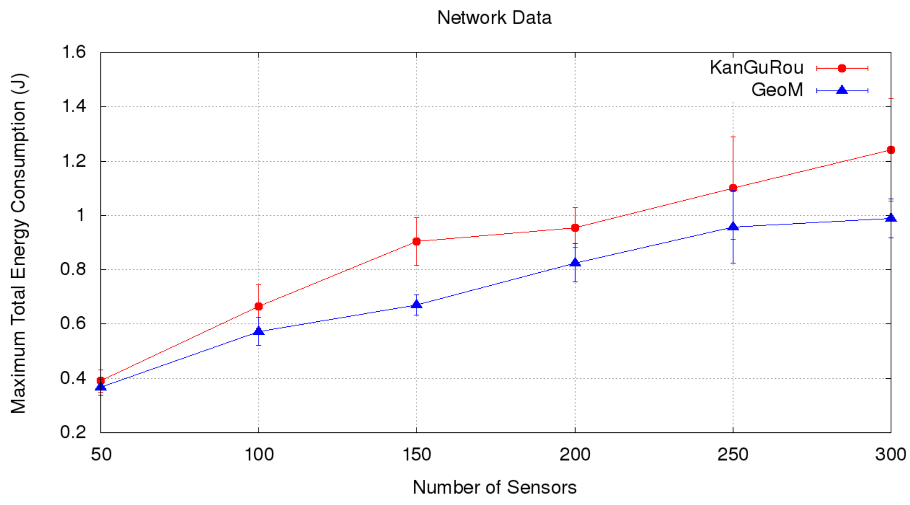

Fig. 5. Maximum Energy Consumption results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes in networks with void areas.

consequently reducing the maximum energy consumption. The network grow affects GeoM positively, with an increase in the performance gain in relation to KanGuRou. With the increase of the network size, the nodes in void areas may be avoided more easily, since both sink options and the possibility of candidate forwarder intersections are also increased.

We can see in figure 6, for the scenario with 300 sensor nodes, that GeoM and KanGuRou have a similar behavior in terms of energy consumption over time. However, GeoM has a slightly better curve compared to KanGuRou, reaching the end of the simulation with a level of energy consumption that represents less than 80% of the energy consumed by KanGuRou.

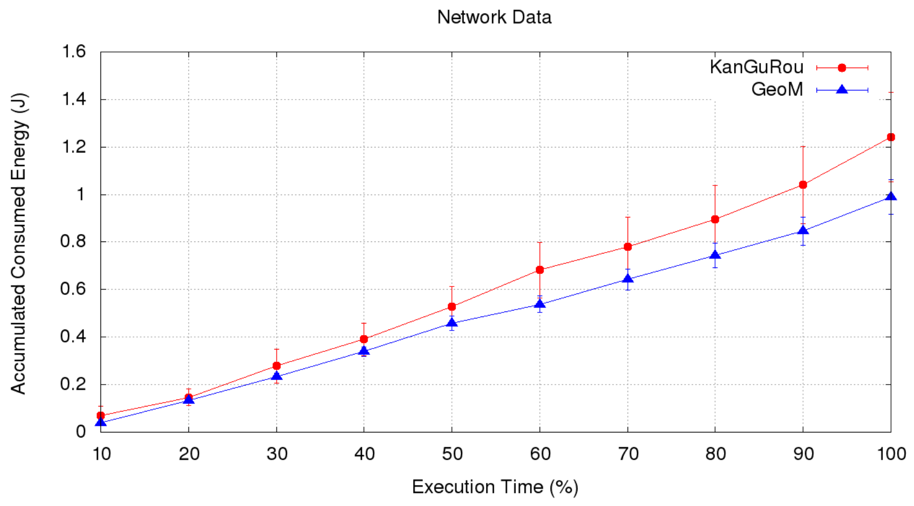

Fig. 6. Evolution of the maximum consumed energy. Results for 300 sensors and 30 sinks (network with void areas).

The distribution of the nodes in terms of energy consumption at the end of the simulation presented in figure 7 shows that GeoM manages to balance the energy consumption, even for the networks with void areas. This may represent an increase of the network lifetime in relation to KanGuRou. If the initial battery level of the nodes was set to $1J$, at the end of the execution, KanGuRou would already have nodes with depleted batteries, while all nodes for GeoM would still be alive.

In terms of latency, GeoM presents a performance gain of around 10% in relation to KanGuRou, as displayed in figure 8. This can be explained by the void announcements strategy. When packets enter in void areas, the paths become longer as a consequence of the void handling technique to exit the void area. However, with the nodes announcing their void situation, neighbors try to forward packets to other directions, avoiding the problematic area.

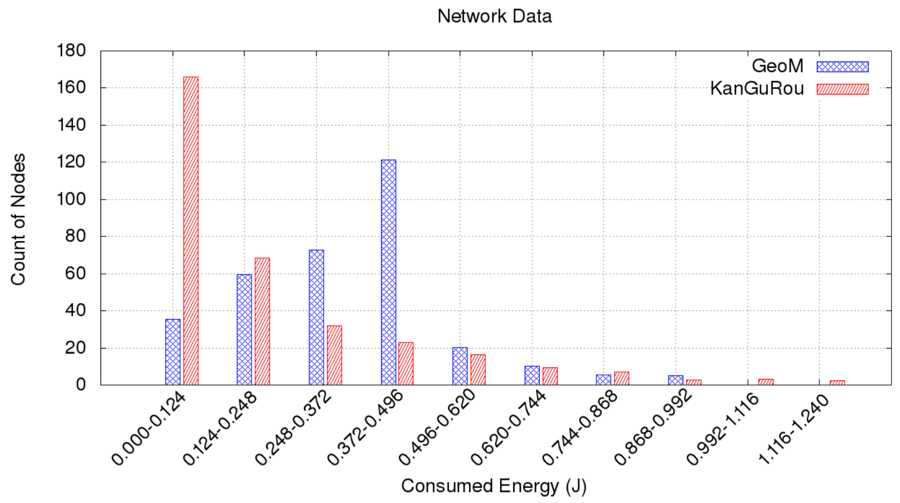

Fig. 7. Final State of the Network with the distribution of nodes in terms of consumed energy. Results for 300 sensors and 30 sinks in networks with void areas.

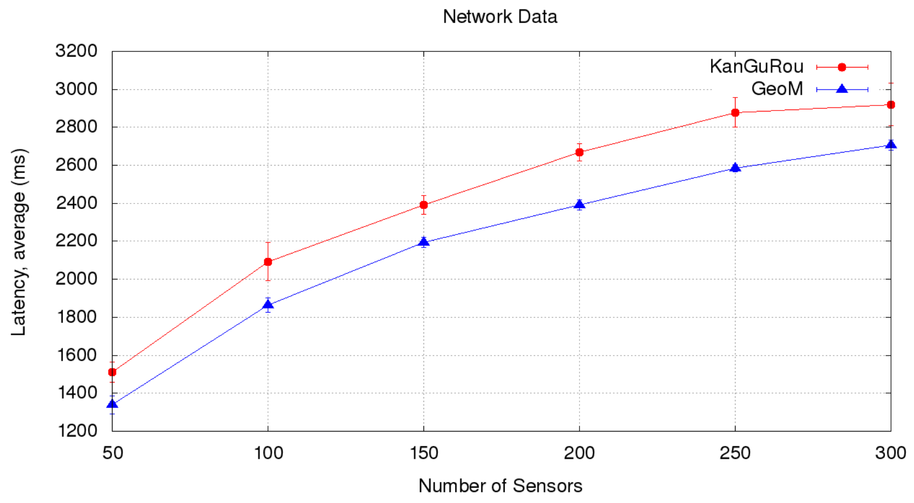

Fig. 8. Average Latency results with sensor varying from 50 to 300 nodes and sinks at 10% of the sensor nodes (network with void areas).

6 Conclusion

This paper presented a new Geographic Multicast Routing solution for Wireless Sensor Networks with multiple sinks. Our strategy makes use of weighted metrics to establish the list of forwarder candidates, and search for intersections among the sinks and candidates in order to avoid duplications. The main goal was to find a balance between Latency and Network Lifetime optimizations. We tested our solutions through simulations against another geographic routing strategy capable of forwarding packets to multiple sinks. The simulation results indicate that our solution has an overall better performance than the existing protocol, with maximum gains of approximately 11% for Latency and 54% for Maximum Energy Consumption.

As future work, we plan to execute real-life experiments for GeoM and KanGuRou using the testbed from the FIT IoT-LAB [7] and evaluate the performance of the solution under real conditions.

Acknowledgments. This work is partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq). Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

References

1. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems. pp. 307–320. ACM (2006)
2. Buratti, C., Conti, A., Dardari, D., Verdone, R.: An overview on wireless sensor networks technology and evolution. Sensors **9**(9), 6869–6896 (2009)
3. Chakraborty, S., Chakraborty, S., Nandi, S., Karmakar, S.: Fault resilience in sensor networks: distributed node-disjoint multi-path multi-sink forwarding. Journal of Network and Computer Applications **57**, 85–101 (2015)
4. Chen, D., Varshney, P.K.: A survey of void handling techniques for geographic routing in wireless networks. IEEE Communications Surveys & Tutorials **9**(1), 50–67 (2007)
5. Ciciriello, P., Mottola, L., Picco, G.P.: Efficient routing from multiple sources to multiple sinks in wireless sensor networks. In: European Conference on Wireless Sensor Networks. pp. 34–50. Springer (2007)
6. Contiki OS: The Open Source OS for the Internet of Things, http://www.contiki-os.org/, accessed: 2018-03-20
7. Facility, F.I.T.: FIT IoT-LAB, https://www.iot-lab.info/, accessed: 2018-03-20
8. Gao, D., Lin, H., Liu, X.: Routing protocol for k-anycast communication in rechargeable wireless sensor networks. Computer Standards & Interfaces **43**, 12–20 (2016)
9. He, X., Kamei, S., Fujita, S.: Autonomous multi-source multi-sink routing in wireless sensor networks. Information and Media Technologies **7**(1), 488–495 (2012)

10. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: 33rd Annual Hawaii International Conference on System Sciences (HICSS). pp. 1–10. IEEE (2000)
11. Heissenbüttel, M., Braun, T., Bernoulli, T., WäLchli, M.: BLR: beacon-less routing algorithm for mobile ad hoc networks. Computer communications **27**(11), 1076–1086 (2004)
12. Kim, D., Wang, W., Wu, W., Li, D., Ma, C., Sohaee, N., Lee, W., Wang, Y., Du, D.Z.: On bounding node–to–sink latency in wireless sensor networks with multiple sinks. International Journal of Sensor Networks **13**(1), 13–29 (2013)
13. Mitton, N., Simplot-Ryl, D., Stojmenovic, I.: Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks. In: 28th Annual IEEE Conf. on Computer Communications (INFOCOM). pp. 2691–2695 (2009)
14. Mitton, N., Simplot-Ryl, D., Voge, M.E., Zhang, L.: Energy efficient k-anycast routing in multi-sink wireless networks with guaranteed delivery. In: International Conference on Ad-Hoc Networks and Wireless. pp. 385–398. Springer (2012)
15. Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., Voigt, T.: Cross-level sensor network simulation with COOJA. In: Proceedings of 31st IEEE Conference on Local Computer Networks. pp. 641–648. IEEE (2006)