Geo-casting of Queries Combined with Coverage Area Reporting for Wireless Sensor Networks

L.F.W. van Hoesel[a,*], A. Erman-Tüysüz[b], A. Dilo[b], P.J.M. Havinga[b]

[a] *Ambient Systems B.V., Coloseum 15d, NL-7521 PV Enschede, the Netherlands*
[b] *Faculty of Electrical Engineering, Computer Science and Mathematics, University of Twente, Postbus 217, NL-7500 AE Enschede, the Netherlands*

Abstract

In order to efficiently deal with queries or other location dependent information, it is key that the wireless sensor network informs gateways what geographical area is serviced by which gateway. The gateways are then able to e.g. efficiently route queries which are only valid in particular regions of the deployment. The proposed algorithms combine coverage area reporting and geographical routing of queries which are injected by gateways. The combined solution is evaluated in terms of computational complexity and performance compared with existing geo-casting protocols.

Keywords: Wireless sensor networks, Geo-casting, Geographical routing, incremental convex hull

1. Introduction

During the lifetime of a wireless sensor network (WSN), the interest in collected data may change. For example, this is likely in environmental

*Corresponding author
Email addresses: lodewijk.vanhoesel@ambient-systems.net (L.F.W. van Hoesel), a.tuysuz@ewi.utwente.nl (A. Erman-Tüysüz), a.dilo@ewi.utwente.nl (A. Dilo), p.j.m.havinga@ewi.utwente.nl (P.J.M. Havinga)

monitoring applications where the sensor network is used as data collection tool by many researchers, who each might be interested in different data sets [1]. Therefore, it is useful to be able to specify to the network, which measurement type (e.g. averaged results) should be used, from which sensors and at what rate data should be collected. The sensor nodes then adjust their operation accordingly.

One way to adjust the data set that a sensor node reports to a central gateway is to reprogram the nodes with updated firmware code. Levis et al. [2] let nodes transmit the version number of their firmware at a slow pace. When they (locally) detect that a neighbouring node has old firmware, neighbours with more recent firmware automatically start update procedures. A similar scheme, Deluge, is presented by Hui et al. [3]. However, in most cases complete reprogramming is too expensive in terms of energy consumption. Reijers et al. [4] propose a reprogramming mechanism based upon efficiently transferring code changes. The program code on the node is basically patched by a change script, executed on the nodes.

It is questionable whether reprogramming is a good strategy to specify a change in sensor data that needs to be reported to a central gateway, especially when this changes frequently. Instead of thinking of the network as executing pre-programmed tasks, one might also consider the sensor network as a distributed database, which can be queried for sensor readings. In this setting, the nodes are pre-programmed with a query interpreter, which stores and parses incoming queries, and changes the node's behaviour accordingly [5]. Madden et al. provide in [6] a detailed description of TinyDB, the format of TinyDB queries and filtering functions that can be established with these

queries.

This work is in particular focused on the routing of queries (or other message types) in the wireless sensor network that contain filtering on location i.e. the query is only valid in particular regions in the wireless sensor network. Higher efficiency can be achieved when the geographical filtering is applied during the message routing process, instead of flooding queries to all nodes in the network and then let the nodes apply the wished geographical filtering. Obviously, the latter may involve much more communication between nodes, which is typically a large source of energy expenditure for wireless sensor nodes [7].

Higher efficiency may not only be achieved in the wireless sensor network, but potentially also in a powerful mobile ad-hoc network (MANET), which interconnects gateways that each communicate with a subset of wireless sensors. Gateways collaborate with other MANET enabled devices to extract contextual information from the sensor network by inserting queries. These queries inform the wireless sensors which information needs to be delivered to the gateways and are only inserted into the (local) sensor network if relevant. The proposed geographical routing mechanism delivers as side product a coverage area description of the wireless sensor network. This description provides context information to applications of the WSN and can potentially lead to more efficient control and use of the wireless sensor network and e.g. the generation of sensible queries by the application.

In this paper, a mechanism is proposed in which (1) the wireless sensor network provides an accurate and up-to-date coverage area description to gateways and (2) the wireless sensor network re-uses the collected coverage

area information to enable geo-casting of location dependent queries and other messages. The latter has a focus on routing of messages injected from a gateway to nodes in the region of interest.

In short, the distributed mechanism works as follows. Sensor readings are generated by sensor nodes according to queries and are —together with position information of the node– encapsulated into messages. These messages are sent to a gateway in the wireless sensor network. The messages potentially need to travel multiple hops in the network before they reach the gateway. Sensor nodes, through which the messages flow, inspect the position information inside the message and use this information to create a local coverage area description, which is stored in the sensor node itself. The local description represents the area which is covered by the sensor node itself and all other nodes from which messages flow through the node on their way to the gateway. At the gateway, the local coverage area description represents the covered area of the subnet served by this gateway. The local coverage area descriptions are continuously updated when nodes receive position information.

When the application injects a location dependent query or other message into the network, the gateway checks if there is a match between the area covered by the sensor network and the area specified in the query. If so, the gateway forwards the query to its child nodes. In their turn, these nodes check if there is a match between their local coverage area descriptions and the query's specified area. If a node finds a match, it again forwards the query to its child nodes. If not, the query is simply not propagated. In this way, the query is routed to the area where it needs to be executed.

This paper is organized as follows. Related work is presented in Section 2. Section 3 discusses the design of distributed coverage area reporting functionality. The section also provides general assumptions used in this work. In Section 4, the geo-casting of location dependent queries is discussed. The proposed combination of coverage area and geographical routing is called *GEOCAST*. Section 5 focuses on implementation aspects on resource-constrained sensor nodes. Throughout this work, the limitations of sensor nodes in terms of computational capabilities and energy reserves are considered. The performance of the proposed routing mechanism is compared with existing work in Section 6. Section 7 provides conclusions.

2. Related work

In traditional (computer) networks, routing of messages is most commonly based on logical addresses of devices. A different routing strategy for wireless sensor networks is described in [8]: geographical routing. Instead of advertising an interest for data, or requesting to establish a route to a certain destination device, nodes use a routing technique based on node coordinates. Nodes are assumed to know their own position and the position of the destination node (i.e. the node where the data needs to be delivered). The idea is that nodes advertise data along with the coordinates where it must be delivered. Nodes closer to the destination node consider themselves candidates for relaying the message. In this section, previous work in geographical routing and geo-casting is discussed.

Face Routing [9, 10] routes packets along faces of planar network graphs by using simple right hand rule and proceeds along the line connecting the

source and the sink. Although it guarantees to reach the destination, it does so with O(n) messages, where n is the number of network nodes, and a simple flooding algorithm already reaches the destination with O(n) messages. Also, it is not competitive with the shortest path algorithm in terms of cost depending on the number of hops between the source and the destination.

Adaptive Face Routing (AFR) [11] is the first algorithm competitive with the shortest path between the source and the destination. It basically enhances Face Routing by the concept of an ellipse-bounding region restricting the searchable area. With a lower bound argument AFR was shown to be asymptotically optimal. On the other hand, AFR is not practical due to its pure face routing concept. For practical purposes there have been attempts to combine Greedy approaches (always send the message to the neighbor closest to the destination) and face routing; for example Greedy Perimeter Stateless Routing (GPSR) [12], however, without competitive worst-case guarantees. There have been some other proposals to combine Greedy routing with face routing like the GOAFR and GOAFR+ algorithms by Kuhn et al. [13, 14], which remain worst-case optimal.

In most of these protocols, the packets are sent from source to a destination position. For some other scenarios like general position-based publish-and-subscribe services, it is also sufficient for some packets (e.g. subscriptions, queries, etc.) to reach any destination currently located in a given area (i.e. geo-casting). Yu et al. propose Geographical and Energy-Aware Routing (GEAR) algorithm [15], which shows how to broadcast a message to all the nodes in a target region. GEAR uses greedy forwarding to forward packets to the nodes that are progressively closer to the centroid of the target

region, whilst trying to balance the energy consumption at the intermediate nodes. Once the message is delivered to the centroid of the target region, it then uses restricted flooding, namely Recursive Geographic Forwarding, to broadcast the message to all remaining nodes in the given region. Instead of using geographical forwarding, GeoTORA [16] uses a unicast (ad-hoc) routing protocol (TORA [17]) to deliver the packet to the region and then floods within the target region.

There are some other protocols based on window spanning infrastructure (WSI) for routing to the specified message window (i.e. destination region). In this approach, the message first is forwarded towards the message window by an end-to-end routing protocol. Once the message reaches the window, an infrastructure within the message window is built along with the message propagation. The method in [18] uses a Greedy technique to find a routing path from message originator to a node N_c located at the center of the messages spatial window. This first part of the routing is similar with the approach used in GEAR. For the routing inside the window, the framework proposed in [18] uses two different approaches namely WinFlood and WinDepth. The WinFlood algorithm consists of a constrained parallel flooding, where a node broadcasts the message to its neighbors only if its own location is inside the messages spatial window. The alternative solution, WinDepth, is based on depth first search policy.

As we have seen from the related works given above, the first step of geocasting is generally based on Greedy approach which cannot guarantee that a routing path to a node in the messages spatial window will be found. Stojmenovic [19] reviews the existing approaches for message delivery to a

destination region, i.e. geocasting. Three approaches that guarantee delivery in static sensor networks are discussed in detail: (i) face traversal scheme based on depth-first search of the face tree, (ii) traversal of all faces that intersect the border of the geocasting region, and (iii) entrance zone (i.e. the set of points that are at smaller distance than the transmission radius R from the destination region) multicasting-based geocasting. These algorithms mainly solve the routing hole problem in sparse networks in order to guarantee the delivery of messages to the target region. However, face traversal has considerable communication overhead as we discussed previously, so these approaches cause unnecessary overhead in dense networks. An adaptation to traversal of faces intersecting the target region (called $GFPG^*$), which achieves delivery guarantee in sparse networks and reduces the additional overhead of face traversal scheme in dense networks, is discussed in [20]. In the $GFPG^*$ algorithm, each node inside the geocasting region divides it radio range into four equal partitions. If there is at least one neighbor in each partition, it is assumed that there is no gap around this node. Thus, this node will not send perimeter packets (i.e. initiate face traversal) and will send only the geocast message inside the target region. If a node has no neighbor in a partition, it enters the perimeter mode and uses right-hand rule to send perimeter packets.

The geocast routing protocols discussed above are non-flooding based approaches, meaning other routing protocols are used to reach the target region instead of flooding, e.g. greedy forwarding, ad-hoc routing. Regional flooding may still be used inside the target region. The authors in [21] also discuss *directed flooding* based geocast routing protocols. Directed flooding

tries to limit the message overhead and network congestion of naive flooding by defining a forwarding zone, which consists of a subset of all network nodes. The forwarding zone includes at least the sender of the geocast message and the target region of the message. It should also include a routing path between source node and target region. Otherwise, protocols either have to increase the size of the forwarding zone or fall back to simple flooding. An intermediate node forwards a message only if it belongs to the forwarding zone. Directed flooding based geocast protocols [22, 23, 24] differ in how they define the forwarding zone. In Location-Aided Routing (LAR) [22], the forwarding zone is the smallest rectangle that includes the sender node and the target region. Voronoi diagram approach in [23] defines the forwarding zone as follows: a neighbor of a sender node belongs to the forwarding zone if and only if it is closest in the direction of the destination region. GeoGRID [24] partitions the network into logical grid cells and a single elected node close to the center of each grid cell is responsible for propagating geocast packets to neighboring cells.

Recently there are also proposals for geocasting of a message to several geocast regions. The authors in [25] combine clustering and multi-geocasting for delivery guarantee to multiple target regions in WSN. In this work we assume that each query packet specifies only one target region. We do not consider multiple target regions (i.e. multi-geocasting) for a single query packet in this work. It is assumed that when a sink node needs to send the same query to different target region, it has to generate separate query packets for each target region.

Table 1 presents a comparison of all discussed geocasting protocols. The

Table 1: Comparison of geocasting protocols (TR: Target Region, FZ: Forwarding Zone)

Protocol	Path Strategy	Routing towards TR	Routing inside TR	FZ
GEAR (2001)	Unicast	Greedy Forwarding	Flooding	-
GeoTORA (2003)	Unicast	TORA ad-hoc routing	Flooding	-
WSI (2004)	Unicast	FullFlood/GreedyDF	WinFlood/WinDepth	-
GFPG* (2006)	Unicast	Greedy Forwarding wih face routing	Traversal of faces intersecting TR	-
LAR (1998)	Multicast	Directed Flooding	Regional Flooding	Rectangle
GeoGRID (2000)	Multicast	Directed Flooding	Regional Flooding	Rectangle
Voronoi (2003)	Multicast	Directed Flooding	Regional Flooding	Polygon

main differences between the protocols are observed either in the first phase, which is flooding based or non-flooding based for routing towards target region, or in the second phase that is the routing inside the specified message window. However, our approach uses a different technique, which does not make a distinction between approaches on two phases. Coverage area descriptions are used in the first phase of routing to forward the packets from source to the given area. It is again the coverage area description that is used in the second phase, transmitting packets to nodes inside the target region. In the performance evaluations we compare our approach with GEAR [15]. All the non-flooding based geocasting approaches take GEAR as a basis. The other group of geocasting approaches, based on directed flooding, perform worse compared to GEAR [21]: GEAR achieves higher delivery success ratio and lower message overhead than the other directed flooding protocols with

rectangle and cone forwarding zones. The following sections describe our approach in detail.

3. Coverage area reporting

The first step in the geographical routing of location dependent queries is to establish a notion of what area is covered by the WSN. Then, the actual geographic situation is compared with the target geographical zone in queries and routing decisions are executed accordingly. In this section we discuss the design for distributed coverage area reporting. In the distributed approach of establishing a description of WSN coverage area per gateway, nodes keep track of partial information of the coverage area. In this way, gateways are efficiently informed of the coverage areas, while the amount of information each node needs to store, transmit and receive is limited. Throughout this work, the limitations of sensor nodes in terms of resources (Section 5) are a driving force behind design choices.

3.1. Approach and assumptions

We assume that each of the nodes in the wireless sensor network has the ability to obtain an estimate of its position. This can be either by localization mechanisms [26, 27, 28, 29, 30], GPS or by other means (e.g. [31]). Whenever a node publishes information, it is augmented with the current position of the node.

By the term *coverage area*, we understand the geographical area in which the sensor nodes are deployed. It is important to note that this is not equal to *information coverage* as defined by Wang et al. [32]: the coverage area indicates points where events are detected, but it does not define where

events can be located to get detected. Although information coverage is important, it is not considered in the proposed routing mechanism, to keep the proposed routing generic and independent of sensor types. In general, sensor horizons are different for each sensor type or sensor implementation. It may be dependent on the orientation of the sensor and sensor nodes may be equipped with multiple different types of sensors [33].

The coverage area description, which the proposed mechanism delivers as context information to the application of the wireless sensor network, is actually only a approximate description: it provides the convex hull of the locations where nodes are deployed. Edelsbrunner et al. present so called α-shapes [34, 35], which construct an intuitive shape based on a set of points, where the parameter α determines the crudeness of the resulting shape. This class of shapes is a good candidate for representation of the wireless sensor network coverage area, if the α-parameter is correctly tuned with respect to the transmission range of nodes.

In this work coverage areas are represented by convex hulls of the node locations (the convex hull is an α-shape with α approximating 0). With this 'crude' coverage area description detail is lost, e.g. holes in the wireless sensor network deployment. Many geographical routing protocols (Section 2) need to take special precautions to ensure that messages are not stuck at holes in the deployment. However, the coverage area information collection described in this section is tightly related to information flow from sensors towards the gateways. The information flow in this direction is defined by routing trees, one for each gateway, which implicit connectivity assures avoidance of networks holes. We assume that links between nodes are bi-directional.

Throughout this work, the following is assumed in the wireless sensor network. One or more gateways are deployed with the wireless sensors and a wireless sensor node is logically grouped with one gateway to balance the load across the gateways e.g. using [38, 39, 40]. Topology constraints, such as connectivity, and load balancing are taken into consideration. Basically, the routing strategy of the wireless sensor network determines which node reports to which gateway. This work assumes that a routing tree is present to route information efficiently to a selected group gateway e.g. the work in [41]. We use the definitions *parent node* and *child node* to indicate node positions in the routing tree. A child node has selected the parent node as intermediate node in order to get messages towards a gateway. Nodes store the logical address of their parent node, but do not need to keep track of their (possibly many) child nodes. The reason for this is to keep the minimal necessary information for routing in order to minimize storage of a sensor node. The protocol that we propose for routing of queries is independent of the mechanism used to partition the network between the different gateways, and the way the routing trees are built. In our implementation we use Voronoi decomposition based on hop-count metric for network partitioning, and the shortest path routing metric to create the trees.

An overview of the presented approach of establishing coverage area descriptions is depicted in Figure 1. Nodes keep track of coordinates that are either included in messages carrying sensor data, or are explicitly transmitted as described in Section 3.4. Using the received coordinate information, a node creates its local coverage area description, represented as a convex hull. Periodically, the local convex hull is transmitted to the parent node

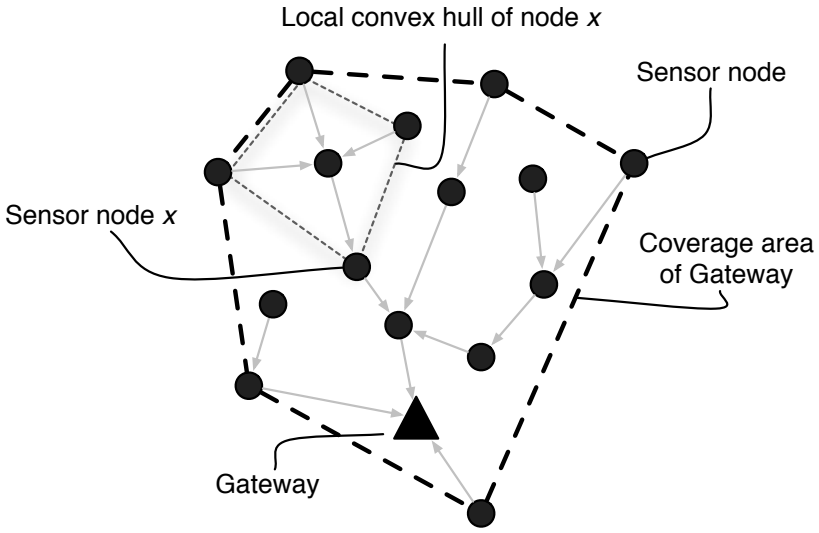


Figure 1: Example network with coverage area (i.e. local convex hull stored in the gateway) and a local convex hull stored in a sensor node

that merges the received convex hull with its local coverage area description. A parent node maintains a convex hull that envelopes the node itself and all its descendant nodes in the tree (see the local convex hull of node x in Figure 1). The coverage area of a gateway is the convex hull of all the sensor nodes served by this gateway (see Figure 1). Optionally, a convex hull is reduced using some form of compressing before transmitting in order to limit memory usage by the algorithm and energy consumption by reducing the size of transmitted/received coordinate list.

3.2. Definitions

Let $\mathbf{C^0} = \{c_0, c_1, \ldots, c_i\}$ be a set of locations, where each location c is a two dimensional coordinate $(c^{(x)}, c^{(y)})$. Let the function $\mathcal{CH}(\mathbf{C^0}) = \mathbf{H}$ create

a minimal (ordered) set of coordinates $\mathbf{H} \subseteq \mathbf{C^0}$ that envelops the coordinates in set $\mathbf{C^0}$. $\mathbf{H}$ is called the convex hull of the coordinate set $\mathbf{C}^0$. We assume that the coordinates in $\mathbf{H}$ are ordered such that the convex hull encompasses the coordinate set $\mathbf{C^0}$ counter clockwise. We denote $|\mathbf{H}|$ as the number of elements in the set $\mathbf{H}$. Note that $|\mathbf{H}| \leq |\mathbf{C}^0|$.

Many methods are proposed in literature to create the convex hull of a set of locations e.g. [42]. Typically, these algorithms operate on a set of locations and produce a convex hull, but do not consider addition of locations once the convex hull has been created. In the following section, we present an algorithm that constructs and maintains the (local) convex hull of a node when sensor readings with coordinate information or periodically transmitted local convex hulls are received by the node. In fact, the algorithm implements a merge function $\mathcal{CH}(\mathbf{H}, \mathbf{C}) = \mathbf{H}'$, where $\mathbf{C}$ can be a single location, a set of locations or a convex hull, with $\mathbf{H}' \subseteq \mathbf{C} \cup \mathbf{H}$. The presented algorithm relies on the fact that the convex hull $\mathbf{H}$ is stored as a counter clockwise ordered set and, when merging, the ordering is kept.

Let $\overrightarrow{h_k h_{k+1}}$ be the vector connecting location h_k with h_{k+1} of the convex hull $\mathbf{H}$. When the indices are larger than the size of the set e.g. when $k + 1 > |\mathbf{H}|$, the modulo with the set size is meant. To determine if the coordinate c is on the left side of a vector, we make use of the right hand-rule, by checking the orientation of the cross product $\overrightarrow{h_k h_{k+1}} \times \overrightarrow{h_k c}$. In the two-dimensional case, the cross product $\overrightarrow{h_k h_{k+1}} \times \overrightarrow{h_k c}$ is equivalent to

$$d = (h_k^{(y)} - h_{k+1}^{(y)})c^{(x)} + (h_{k+1}^{(x)} - h_k^{(x)})c^{(y)} + h_k^{(x)}h_{k+1}^{(y)} - h_k^{(y)}h_{k+1}^{(x)} \qquad (1)$$

Then, the coordinate c is on the left of line segment $\overrightarrow{h_k h_{k+1}}$ if the result of Equation (1) is positive: $d > 0$. This check will later on be used to see if a

coordinate is inside a convex hull.

Finally, we define p_i to be the position of node i in the wireless sensor network.

3.3. Constructing local coverage area descriptions

In this section, we discuss how a node constructs a local coverage area that describes the area covered by the node itself, its child nodes and other descendant nodes. To create a local coverage area description, a node inspects all messages that 'flow' through the node towards the designated gateway. The location information inside these messages is used as input for the construction algorithm.

Let $\mathbf{H_i}$ be a convex hull representing the (local) coverage area description of a wireless sensor node or gateway i. The coordinate set $\mathbf{H_i}$ is always ordered such that it describes the convex hull counter clockwise. Initially, $\mathbf{H_i} = \{p_i\}$ contains the coordinate of the node itself, however, during the update process described below, the coordinate of the node itself might be removed from $\mathbf{H_i}$.

Let $\mathbf{C}$ be the set of coordinates that a node or gateway receives ($\mathbf{C}$ is either a single coordinate which is extracted from a sensor reading flowing through the node or a received convex hull from a child node). If $\mathbf{C}$ is an empty set, our algorithm applies no changes to $\mathbf{H_i}$. Otherwise, per coordinate in the set $\mathbf{C}$ the following procedure is executed:

1. Define c_j as current coordinate to investigate from the set $\mathbf{C}$ ($0 \leq j \leq |\mathbf{C}| - 1$). If this coordinate is already present in the set $\mathbf{H_i}$, move on to the next coordinate. We investigate per coordinate if it is inside $\mathbf{H_i}$. If not, $\mathbf{H_i}$ is adjusted such that it envelops the coordinate as well.

2. Let $n = |\mathbf{H_i}|$ be the number of coordinates in the local convex hull:

   - **One coordinate ($n = 1$)** — Add the coordinate to $\mathbf{H_i}$ and order the coordinates such that the coordinate with lowest y-value is first in the set.
   - **Two coordinates ($n = 2$)** — Use Equation (1) to check if c_j is on the left of the line segment $\overrightarrow{h_0 h_1}$ (see Section 3.2). If so, put the coordinate at the third position in the convex hull $\mathbf{H_i}$, otherwise insert the coordinate between h_0 and h_1 in $\mathbf{H_i}$.
   - **More coordinates ($n > 2$)** — Check for each line segment $\overrightarrow{h_0 h_1}$, $\overrightarrow{h_1 h_2}$, ..., $\overrightarrow{h_{n-1} h_n}$, $\overrightarrow{h_n h_0}$ if the coordinate c_j is on the left of the line segment. If so, the coordinate is enveloped by the convex hull $\mathbf{H_i}$; continue with the next line segment.

     If c_j is not on the left of a line segment, then record the starting coordinate of the line segment as begin point b. Continue with the next line segments until c_j is left of the line again. Remove all coordinates from b until the current line segment and insert c_j instead.

The above procedure is applied when a node receives sensor readings augmented with position information or when local convex hulls are explicitly propagated from child nodes. Next, we describe tasks that nodes need to execute periodically to keep their local coverage area up to date in dynamic networks.

3.4. Removing coordinates from local coverage area descriptions

Due to dynamics in network topology, the local convex hull maintained in a node can contain coordinates that do no longer reflect the actual coverage area of the node, its children and other descendants e.g. this might be the case when a node dies/fails or nodes are mobile. To keep the local convex hull accurate, a time out mechanism is required to remove old coordinates from the local convex hull. Also, significant changes in the routing tree may be used as trigger to recreate a local coverage area description. The latter topic is left for future work.

Nodes store a timestamp for each individual coordinate in their local convex hull $\mathbf{H_i}$. The timestamp of a particular coordinate is reset when a node receives a message containing the coordinate. But when a coordinate has not been reinforced within the time out interval, it is removed from the local convex hull and is therefore also not propagated to the parent node. The time out information is never propagated to parent nodes.

A suitable time-out interval needs to be determined according to the level of mobility in the network, however, it must not be shorter than the interval at which nodes produce sensor readings, otherwise coordinates are removed from the local convex hulls before they are reinforced. If topology changes are frequent, the time-out interval should be short to ensure up-to-date coverage area descriptions. The effects of node mobility are discussed in Section 6.

Periodically i.e. once per time-out interval, a node applies the algorithm described in Section 3.3 to check if its own position p_i must be added to the local coverage area description. This action also ensures that a potential time out on the own coordinate is prevented. Note that p_i is always either

inside the area bounded by $\mathbf{H_i}$, or a coordinate in the set $\mathbf{H_i}$ or on one of the line segments represented by the convex hull $\mathbf{H_i}$.

3.5. Compression of coverage area descriptions

The proposed mechanism for distributed coverage area reporting requires that nodes (periodically) propagate the convex hull that describes the local coverage area to parent nodes. Obviously, message sizes grow with the number of coordinates that are part of the convex hull. Consequently, more accurate, but larger coverage area descriptions result in higher energy expenditure of the nodes. Therefore, compression (i.e. approximation of the convex hull with a smaller coordinate set) is an attractive option to limit resource consumption, such as energy and bandwidth. It is important to note that a convex hull is already a minimum set by itself.

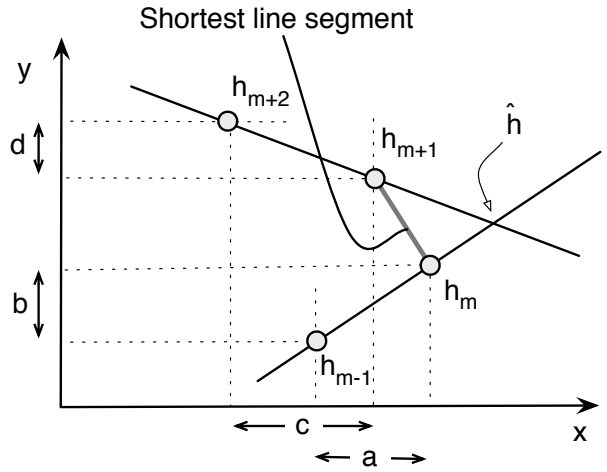


Figure 2: Compression of the convex hull removes short line segments from $\mathbf{H}$ by adding coordinates at the intersection of segments before and after the short segments

The compression algorithm accepts as input a convex hull and a maximum convex hull size $n_{max} > 3$. Until the convex hull has been reduced to maximum size n_{max}, the algorithm finds two coordinates h_m and h_{m+1} which represent the shortest line segment in the convex hull with $r_m \neq 0$ (Equation 4). These two coordinates are then removed from the convex hull $\mathbf{H}$ and are replaced with one coordinate $\widehat{h}$, such that h_m and h_{m+1} are both on the line segments $\overrightarrow{h_{m-1}\widehat{h}}$ and $\overrightarrow{\widehat{h}h_{m+2}}$, respectively (Figure 2). The coordinate $\widehat{h}$ is positioned at the intersection of the line passing through h_{m-1}, h_m and the line through h_{m+1}, h_{m+2}. It is calculated as follows [43]:

$$\widehat{h}^{(x)} = \frac{1}{r_m} \begin{vmatrix} h_{m-1}^{(x)} h_m^{(y)} - h_{m-1}^{(y)} h_m^{(x)} & h_{m-1}^{(x)} - h_m^{(x)} \\ h_{m+1}^{(x)} h_{m+2}^{(y)} - h_{m+1}^{(y)} h_{m+2}^{(x)} & h_{m+1}^{(x)} - h_{m+2}^{(x)} \end{vmatrix} \tag{2}$$

$$\widehat{h}^{(y)} = \frac{1}{r_m} \begin{vmatrix} h_{m-1}^{(x)} h_m^{(y)} - h_{m-1}^{(y)} h_m^{(x)} & h_{m-1}^{(y)} - h_m^{(y)} \\ h_{m+1}^{(x)} h_{m+2}^{(y)} - h_{m+1}^{(y)} h_{m+2}^{(x)} & h_{m+1}^{(y)} - h_{m+2}^{(y)} \end{vmatrix} \tag{3}$$

with

$$r_m = \begin{vmatrix} h_{m-1}^{(x)} - h_m^{(x)} & h_{m-1}^{(y)} - h_m^{(y)} \\ h_{m+1}^{(x)} - h_{m+2}^{(x)} & h_{m+1}^{(y)} - h_{m+2}^{(y)} \end{vmatrix} \tag{4}$$

Note that $r_m = ad - bc$ according to the definitions in Figure 2. If the intersection point $\widehat{h}$ does not exist i.e. the two lines $h_{m-1} \leftrightarrow h_m$ and $h_{m+1} \leftrightarrow h_{m+2}$ are parallel and hence have equal slopes $b : a = d : c$, then follows $r_m = 0$. When this is the case for a shortest line segment, it is skipped by the compression algorithm. Consequently, convex hulls with $|\mathbf{H}| = 4$ having parallel opposite line segments, cannot be further reduced. However, a reduction to a triangle is possible in other cases. Hence, we limit $n_{max} > 3$. Since $\widehat{h}$ is on the exact intersection point of the line segments before and after the line segment which is removed, the substituted coordinates are both on

the new line segments originating from $\widehat{h}$. As a result, the coordinates h_m and h_{m+1} are never added to the compressed convex hull by the algorithm described in Section 3.3, because Equation (1) results in $d = 0$, while $d > 0$ is the criterion to add a coordinate to the local convex hull.

In this work, reduction is only applied when a copy of the local convex hull is forwarded to parent nodes. Nodes maintain the actual convex hull in memory to use detailed information for the geographical routing decisions (Section 4). Obviously, the larger the local coverage area descriptions, the more memory is consumed by the uncompressed convex hull, more processing is needed to apply the merging of coordinates (Section 3.3) and the energy expenditure of nodes will be larger. Therefore, it could be a trade-off to apply compression also to the local convex hull. However, when reduction is only applied on copies forwarded to parent nodes, the time out mechanism of coordinates remains functional without having to e.g. match coordinates to substituted coordinates. In any case, the parent node works with the compressed version. This implies that compressed local convex hulls need to be forwarded to parent nodes within the retention period of coordinates to ensure that substituted coordinates are not removed due to time out.

4. Geographical routing based on local coverage area descriptions

With the above described algorithms, the WSN gateways are informed of a 'crude' description of their coverage area. Next, this information can be used to optimize handling of position dependent information e.g. gateways can use the information whether a certain query is relevant for their coverage area. If not, the gateway can decide to discard the query without inserting

it in the WSN, which in the end saves energy and prolongs the lifetime of the wireless sensor network. Geographical routing of location dependent queries is discussed in this section. Gateways and sensor nodes implement identical functionality regarding the forwarding of queries. Queries are always forwarded from parent nodes to child nodes to get delivered to the area that is specified in the query.

First, we have a closer look at the structure of location dependent queries. We assume that these queries consist of two parts: (1) a description of the area in which the query must be executed, and (2) a command sequence (e.g. sensor types, sample rates, critical thresholds, aggregate functions etc). This work is mainly concerned with the first part of the query. Let $\mathbf{R} = \{r_0, r_1, \ldots, r_n\}$ be the coordinate set describing the region of interest extracted from the query, $\mathbf{H_i}$ the local coverage area description of node i and p_i the (estimated) position of node i.

Upon receiving a query, a node analyses $\mathbf{R}$ and takes two decisions: (1) execute decision (to find out if the node is within the region of interest and needs to execute the query) and (2) forwarding decision (to find out if the node has child nodes or further descendants in the region of interest). Both decisions use $\mathbf{R}$ as input together with p_i and $\mathbf{H_i}$, respectively (Figure 3). If these decisions are translated to addressed based routing, the execute decision is analogous to check if the node is the final destination of the data packet —and the data is handed over to higher layers in the OSI model– and the forwarding decision is analogous to finding a path to the final destination. However, there are slight differences, as we describe in the next sections.

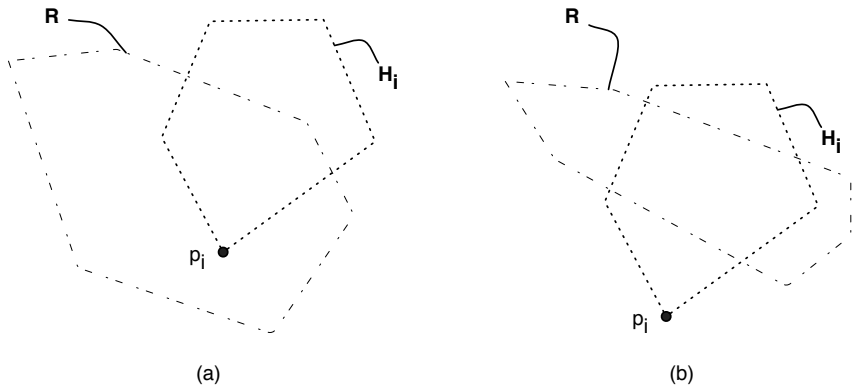


Figure 3: Routing decisions: (a) node i executes query if p_i inside $\mathbf{R}$ and (b) node i forwards query to children if $\mathbf{H_i}$ overlaps with $\mathbf{R}$

4.1. Execute decision

When a node receives a query, it decides if the query is valid for it and, if so, the query is stored and executed until it expires. The *execute decision* basically checks if the node is inside the region of interest i.e. if point p_i is inside the polygon $\mathbf{R}$.

The point-in-polygon problem is a well known problem in computational geometry and many solutions and implementations have been proposed [44, 45]. In general, these algorithms need complex geometrical operations [44]. Looking from a node implementation perspective, it is beneficial to make the assumption that the region of interest in the query $\mathbf{R}$ is a *convex* polygon. This assumption is not strictly necessary, but it reduces the complexity of the implementation, because of properties of a convex polygon [44]. Also, note that in the case of a convex polygon $\mathbf{R}$, parts of the algorithm in Section 3.3 can be reused to check if the coordinate p_i is inside $\mathbf{R}$: the node needs to

check for every line segment if the coordinate p_i is left of the line. The query needs to be executed when p_i is left of all the line segments of $\mathbf{R}$.

When the area of interest cannot be captured with a convex polygon, we assume that multiple queries are generated to cover the complete area e.g. according to the algorithms presented in [46, 47] to decompose polygons into multiple convex parts. This functionality can be realized e.g. within the different gateways, which are likely to be more computational powerful than sensor nodes.

4.2. Forwarding decision

With the forwarding/halting decision a device determines if there *might* be child nodes or nodes further down the routing tree that are within the area of interest specified in the query. If there are, the node should forward the query to its child nodes, which in their turn decide if the query needs to be propagated.

Due to our choice to represent the coverage area by a convex hull, a node cannot determine with certainty that there is indeed a node within the polygon $\mathbf{R}$, because details on node positions are lost for those located within the convex hull. However, a node is able to decide with certainty that further in its part of the routing tree no node is present within the region $\mathbf{R}$. In the later case, the node does not forward the query and consequently does not spend energy on transmitting. In this way it saves resources from its child nodes i.e. receiving the query by radio and processing it as described in this section.

The forwarding decision is taken based upon $\mathbf{R}$ extracted from the query and the local coverage area description $\mathbf{H_i}$. The local convex hull describes

the area from which data messages flow through a node towards a gateway. Our geographical routing, which performs routing of query messages in the opposite direction, exploits the data flow routing paths summarized in convex hulls. A certain region of interest can be reached by a node, if the region overlaps with the local convex hull of the node.

To check if $\mathbf{R}$ overlaps with $\mathbf{H_i}$, at least one of the following must be true:

- Any of the coordinates $r_0, r_1, \ldots, r_n$ form $\mathbf{R}$ is inside the area represented by $\mathbf{H_i}$. Section 4.1 describes how to verify if a point is in a convex polygon;

- Any of the coordinates $h_0, h_1, \ldots, h_k$ form $\mathbf{H_i}$ is inside the area represented by $\mathbf{R}$;

- Any of the line segments represented by the convex polygon $\mathbf{R}$ intersects with any of the line segments from the convex hull $\mathbf{H_i}$. Section 3.5 describes how to calculate the intersection coordinate of two lines. The operation can be used to verify if line segments intersect by adding a check if the intersection point is within the x and y bounds of both line segments.

If any of the checks results true, the area of interest and the local coverage area overlap (Figure 3(b)). In that case the device forwards the query, and the other checks are omitted. In none of the checks result true, the area of interest does not overlap with the local coverage area. The forwarding of the query is halted.

When a node decides to forward the query, it propagates the query to its child nodes. For this purpose, we introduce a special multicast address

in the wireless sensor that represents restricted flooding to child nodes of a node: a node decides to receive a packet if it carries the multicast address and the packet originates from its parent node. A potential optimization step concerning this aspect is to keep track as parent node which area is covered by which child and only forward the query to relevant child nodes. However, it must be noted that this requires nodes to store more information, and makes the forwarding decision more resource consuming because it must be repeated for all stored local convex hulls of children. We leave this trade-off to our future work.

Note that if a node decides to execute a query (Section 4.1), the node also automatically forwards the query, because of the relation of node's location p_i and its local convex hull (Section 3.4). Of course, the query is only forwarded when a node has reason to i.e. it is not a leaf node in the routing tree. This can easily be verified without maintaining a list of child nodes by inspecting the size of the local coverage area description, $|\mathbf{H_i}| > 1$.

The fact that detail of the convex hulls are reduced by compression does not affect the certainty with which nodes can decide to halt the propagation of a query. However, it is expected that forwarding of queries happens more frequently without nodes being present in the region $\mathbf{R}$, simply because the compression scheme of the convex hulls results in larger areas represented by lesser coordinates in $\mathbf{H}$.

5. Implementation aspects for resource-constrained senor nodes

The presented algorithms to obtain coverage area descriptions and the geo-casting of location dependent queries using coverage areas, have been

designed with resource-constrained sensor nodes in mind. Typically, these platforms use microcontrollers running at 4 to 8 MHz with on-chip RAM and program memory. The non-volatile program memory ranges between 32 kB and 128 kB, while the volatile memory is considerably smaller, ranging from 2 kB to 10 kB. Usually the nodes can access external non-volatile memory to store arbitrary data. In [48] an overview is provided of common sensor node hardware platforms. In this section, we dive into the implications it has to implement the presented algorithms on resource-constrained sensor nodes in terms of computational complexity and memory usage.

Let denote with s_c and s_{ct}, respectively the number of bytes required to store a single coordinate $(c^{(x)}, c^{(y)})$ and a single coordinate with time out information.

5.1. Computational resources

The computational complexity of the algorithms —discussed in the previous sections– have been well studied in literature. In [49], the computational complexity is discussed of incremental construction of a convex hull: $O(n^2)$ for the construction of the complete convex hull. O'Rourke discusses in [50] a method of merging convex hulls with computational complexity $O(n+m)$. The computational complexity of point in convex polygon is $O(n)$ accoording to [51]. In [52], the complexity of determining if two convex hulls overlap is given as $O(log(n+m))$.

In this section, we discuss what (computational) functions and resources are required to maintain the local coverage area description and to make the execute and forwarding decisions. Table 2 summarizes geometrical functions that are required and gives their computational complexity in terms of mul-

tiplications, additions and inversions. Table 3 provides a mapping of the functionality discussed in Sections 3 and 4 to these functions.

The `Distance` function calculates the distance between two coordinates. This function is only used in the compression of the local coverage area description before it is forwarded to parent nodes. Since the function is used in the context of finding a *minimum* distance, it suffices to calculate squared distance as follows $(a^{(x)} - b^{(x)})^2 + (a^{(y)} - b^{(y)})^2$. The next geometric function `Left` is used to determine if a coordinate is left of a line segment. This function is repeatedly used to find if a point is inside a convex polygon and the function is used at several stages of the combined coverage area reporting and geo-casting. The most computational intensive function is the `Intersect` function, which calculates the intersection point of two lines of which two coordinates per line are provided as inputs. This function is used in the compression of the local convex hull and in the halt forwarding decision to determine if two areas overlap.

The three functions `Distance`, `Left` and `Intersect` have been implemented on a sensor node testbed [53] with a Texas Instruments MSP430 microcontroller, which has a clock frequency of 4.6 MHz. Timing information of the three functions has been collected in terms of cycles required on the microcontroller and time to complete the operation (Table 2). The implementation uses $s_c = 4$ bytes to represent coordinates.

Table 3 presents the computational complexity of the functionality required for the combined coverage area and geo-casting in terms of the functions `Distance`, `Left` and `Intersect`. When a node receives a coordinate set $\mathbf{C}$ from a child node to update its local coverage area description, it checks

Table 2: Overview of basic geometrical functions required and their computational complexity

Function	Equation	$*$	$+/-$	$1/.$	Cycles
`Distance(a, b)`		2	3		93 (0.20 ms)
`Left(a, b, c)`	(1)	4	5		135 (0.29 ms)
`Intersect(a, b, c, d)`	(2), (3), (4)	16	15	1	821 (1.78 ms)

(in worst case) for each of the coordinates in $\mathbf{C}$ if it is left of any of the line segments in $\mathbf{H_i}$ (Section 3.3). Hence, at most $|\mathbf{H_i}||\mathbf{C}|$ times the function `Left` is used.

Less trivial is the compressing of the local coverage area description before it is forwarded to a parent node. Note that in every compression round the local convex hull size is reduced with at most one and that $|\mathbf{H_i}| - n_{max}$ rounds take place. Per round, the intersection point of two lines is calculated using `Intersect` and the length of all line segments in the working copy of the convex hull $\mathbf{H_i}$ is calculated using `Distance`. In the first round, there are $|\mathbf{H_i}|$ line segments in the working copy and in the next round $|\mathbf{H_i}| - 1$, until n_{max} line segments are reached. Hence, the `Distance` function is used $\sum_{n=1}^{|\mathbf{H_i}|-n_{max}} |\mathbf{H_i}| - n + 1$ times. The computational complexities of the routing decisions in terms of the functions `Distance`, `Left` and `Intersect` follow straightforward from their description (Section 4).

5.2. Memory requirements

Obviously, node requires $|\mathbf{H_i}|s_{ct}$ bytes to store its local convex hull, $|\mathbf{C}|s_c$ bytes to store temporarily coordinates received from a child node and, at

Table 3: Worst case computational complexity in terms of `Distance`, `Left` and `Intersect` functions

Functionality	Section	`Distance`	`Left`	`Intersect`
Constructing local coverage area	3.3		$\|\mathbf{H_i}\|\|\mathbf{C}\|$	
Compressing local convex hull	3.5	$\sum_{n=1}^{\|\mathbf{H_i}\|-n_{max}} \|\mathbf{H_i}\| - n + 1$		$\|\mathbf{H_i}\| - n_{max}$
Execute decision	4.1		$\|\mathbf{H_i}\|$	
Halt forwarding decision	4.2		$2\|\mathbf{H_i}\|\|\mathbf{R}\|$	$\|\mathbf{H_i}\|\|\mathbf{R}\|$

most, $|\mathbf{H_i}|s_c$ bytes to use as working copy of its local convex hull to create a compressed version. Obviously, storage is also required for a received query and its area of interest $\mathbf{R}$. We consider here only the temporary storage space to analyze the query i.e. $|\mathbf{R}|s_c$ bytes. After analyzing the query, we assume that it is either purged from memory or it will not account to memory consumption in the routing layer, since it is handled by a higher layer. In conclusion, the total memory requirements for the presented routing scheme are $|\mathbf{H_i}|(s_{ct}+s_c)+|\mathbf{C}|s_c+|\mathbf{R}|s_c$ bytes of which $|\mathbf{H_i}|s_{ct}$ bytes are permanently required.

5.3. Discussion

The sizes of the coordinate sets $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{H_i}$ determine to a large extend the computational complexity and the memory requirements of the coverage area reporting and geo-casting (Table 3).

For example, assume $|\mathbf{C}| = n_{max} = 8$, $|\mathbf{R}| = 8$ and $|\mathbf{H_i}| = 16$, then the memory requirements are at most 208 bytes, the merging of $\mathbf{C}$ and $\mathbf{H_i}$ takes approximately 38 ms, compressing $\mathbf{H_i}$ to eight coordinates takes ≈35 ms, while the execute decision takes approximately ≈5 ms and the halt forwarding decision (at most) ≈305 ms, assuming the sensor node platform described in Section 5.1 and $s_{ct} = 5$.

The maximum size of the coordinate set $\mathbf{C}$ can be controlled by the compression parameter n_{max}, however, its effect has two sides. Note that when sensor samples flow through a node towards the gateway, $|\mathbf{C}| = 1$. When setting n_{max} to a low value (i.e. high loss of details in the transmitted local coverage area descriptions), (1) the computational requirements of a child node increase due to the compressing, but (2) the energy-expenditure

Figure 4: Trade-off between compression of local convex hull in energy consumption perspective

for transmissions of local coverage areas and the time required to update a local coverage area description in the parent node are reduced (Figure 4).

The question is: can compression be justified in terms of total calculation time (and the associated energy-expenditure) of parent and child? Obviously, a smaller size of $\mathbf{C}$ results in lesser energy consumption of the parent and child nodes to wirelessly transfer the coordinate set $\mathbf{C}$.

The total time to compress the child's local convex hull $\mathbf{H_j}$ and to merge it with the parents convex hull $\mathbf{H_i}$ is given by (see Table 3)

$$|\mathbf{H_i}| n_{max} * t_l + \frac{1}{2}\left(|\mathbf{H_j}|^2 + |\mathbf{H_j}| - n_{max} - n_{max}^2\right) * t_d + (|\mathbf{H_j}| - n_{max}) * t_i \quad (5)$$

where $3 < n_{max} \leq |\mathbf{H_j}|$ and t_l, t_d and t_i the computation times for the functions `Distance`, `Left` and `Intersect`.

To minimise the computation time, we need to find out where the minimum of Equation (5) lays with respect to the interval $3 < n_{max} \leq |\mathbf{H_j}|$. Because Equation (5) is a hat-shaped parabolic function, we need to find the minimum computation time by finding the position of the *maximum* of the function. Denote n_w as size of $\mathbf{C}$ where the computation time is longest. Due to the symmetrical shape of the parabolic function, there are two relevant cases:

1. The maximum of Equation (5) is located before half of the interval i.e. $n_w < \frac{1}{2}(|\mathbf{H_j}|+4)$. In this case, the minimum combined parent/child computation time is located at $n_{max} = |\mathbf{H_j}|$.
2. The maximum is at or after the middle of the interval $3 < n_{max} \leq |\mathbf{H_j}|$. In this case, the minimum total computation time is located at $n_{max} = 4$. Hence, compression with $n_{max} = 4$ is most energy-efficient.

To find the maximum total computation time, the derivative of Equation (5) is taken with respect to n_{max} and the root is determined. This results in $n_w = \frac{t_l}{t_d}|\mathbf{H_i}| - \frac{t_i}{t_d} - \frac{1}{2}$. Next, we find n_w is beyond or at the half of the interval $3 < n_{max} \leq |\mathbf{H_j}|$ i.e. $n_w \geq \frac{1}{2}(|\mathbf{H_j}| + 4)$. This is the case, when

$$|\mathbf{H_i}| > \frac{\frac{5}{2}t_d + t_i}{t_l - \frac{1}{2}t_d} \tag{6}$$

Hence, when Equation (6) is satisfied, compression with $n_{max} = 4$ is most energy-efficient.

For the sensor node platform described earlier in this section the maximum computation time is when the child's convex hull is compressed to $n_w = 1.45 * |\mathbf{H_i}| - 9.4$ coordinates (for a given size of the parent's local convex hull). Then, for $|\mathbf{H_i}| > 12$, compression to $n_{max} = 4$ results in best performance in terms of calculation time and transmit/receive energy-expenditure.

From the above, we conclude that applying compression in child nodes before transmission is beneficial if Equation (6) is met. However, to make this trade-off effectively in a child node, it must know the size of its parent's coverage area description.

6. Comparative performance evaluation

A comparative performance evaluation of GEOCAST versus GEAR [15] is presented in this section. In [21], the performance of different geocasting protocols is compared and GEAR shows the best performance over all the others. Therefore, we choose GEAR for our comparative performance evaluation. First, we define what metrics are used to compare both protocols in simulation. Then, the details of simulation setup are discussed. We compare the routing performance in different set-ups: static networks, mobile networks, high density and low density. The evaluation results of the different scenarios are presented and discussed.

6.1. Evaluation metrics

The geographical routing protocols are compared in terms of routing accuracy i.e. how well the proposed mechanisms deliver messages to the region of interest defined in a query and in terms of networking performance. Opposite to GEAR, GEOCAST also provides a description of coverage area per gateway in the network. This aspect is left out of the comparison, although it enables more efficient use of a WSN from an application point of view.

We define the following metrics to compare the protocols in terms of routing accuracy :

- **Execution ratio (ER)** — The ratio of nodes that are within the region of interest and execute the query with the total number of nodes within the region of interest. This metric measures how well the routing is able to deliver the query to the region of interest.

- **False execution ratio (FER)** — The ratio between the data sinks that inject the query while none of the nodes in their partitions execute the query and the total number of data sinks. Since there is no partitioning in GEAR protocol, FIR is redefined for GEAR as the ratio between the number of data sinks that inject the query while none of the nodes inside the target region execute the query injected by these sinks and the total number of data sinks. Irrelevant query lead to higher energy expenditure in the WSN partition when injected. We measure this effect with the false injection ratio.

- **False injection ratio (FIR)** — The ratio of data sinks that inject the query while none of the nodes in its partition executes the query with the total number of data sinks. Irrelevant query lead to higher energy expenditure in the WSN partition when injected. We measure this effect with the false injection ratio.

We define the following metrics to compare the protocols in terms of networking performance:

- **Average query delivery delay (AQDD)** — The total time elapsed between the query generation by a gateway and its reception by a sensor node inside the target region, averaged over all gateway-target node pairs.

- **Network load (NL)** — The total number of query packets that are sent from gateways. If more messages need to be transmitted to reach the region of interest, the energy expenditure in the WSN is likely to increase. We measure this effect with network load.

Table 4: Simulation parameters

Parameters	Values for GEOCAST	Values for GEAR
MAC protocol	IEEE 802.11 DCF	IEEE 802.11 DCF
Routing protocol for coverage area reporting	Shortest path routing	-
Routing protocol for queries	Geo-casting	Greedy forwarding + restricted flooding
Tranmission range	250 m	250 m
Target region shape	Circle - radius from 250 m to 800 m	Circle - radius from 250 m to 800 m
Mobility Model	Linear mobility	Linear mobility

6.2. Evaluation setup

We evaluate how our GEOCAST algorithm compares with GEAR, which also handles geographical routing of messages to a region of interest. The details of GEAR are given in Section 2. Table 4 presents the simulation parameters.

Our simulations are based on the network simulator *NS-2* [54] version 2.33. *NS-2* allows us to run wireless simulations in realistic scenarios to validate the design choices of GEOCAST. We use 802.11 as the MAC protocol for our *NS* simulations.

The simulation scenarios use network sizes from 100 to 1000 nodes, which are randomly distributed in a square deployment area. The nodes are configured to have $250m$ transmission range. We generate random topologies,

however, only use connected network topologies (i.e. all nodes can reach each other in one or more hops).

For both GEOCAST and GEAR, the target region is set to a circle centered at the middle of the deployment area. The radius of the circular target region is selected between $250m$ and $800m$, scaled according to network dimensions. The routing decisions of GEOCAST based on point in convex polygon (Section 4) have been adapted to point in circle to match the methodology used in GEAR to describe the region of interest.

In the different scenarios, 3 to 30 gateways are randomly selected in each of the varying network sizes. For GEOCAST, sensor to gateway routing is based on a shortest path algorithm. Sensor nodes send their convex hull definitions to their associated gateway, via parent nodes selected using shortest path routing.

In mobile simulation scenarios, node movement follows the linear mobility model. Node velocities are up to $20m/s$ (72 km/h), which includes walking/running person and vehicular movements. Each data point in the graphs are averaged over 50 simulation runs of different topologies, and we show the mean and 95% confidence interval for the evaluation metrics. The GEOCAST compression of local coverage area descriptions is not used in any of the scenarios.

Average query delivery delay (AQDD) of both protocols is first recorded as the end-to-end delay (the time between query generation and reception of the query in the target region). However, AQDD results show that there is a large difference between the query delivery delays of GEAR and GEOCAST. This is mainly due to the current *NS2* implementation of GEAR as

an extension to existing directed diffusion algorithm [55]. Because an interest message in GEAR passes through a series of filters and agents, a query spends a lot of time between agents resulting in high end-to-end delay. As a result, GEAR delay data is not directly comparable with our protocol's delay data. Therefore, we record the *hop-based transmission delays of a message* (i.e. sum of the times between sending the message by a node and receiving the message by the next node for every hop of the routing path) for GEAR, which excludes the time spent in the routing agents. Only in Figure 6(a), we show also the end-to-end delay of GEAR to present its difference with hop-based delay.

6.3. Static network scenarios

In this section, we evaluate the proposed geographical routing GEOCAST in static networks without any disturbing factors (scenarios 1 to 3). We introduce errors in the position estimates of nodes in scenario 4. Simulations of all scenarios show that the FER of queries in static networks is zero for both GEAR and our protocol GEOCAST. Therefore, FER results are not shown in the graphs.

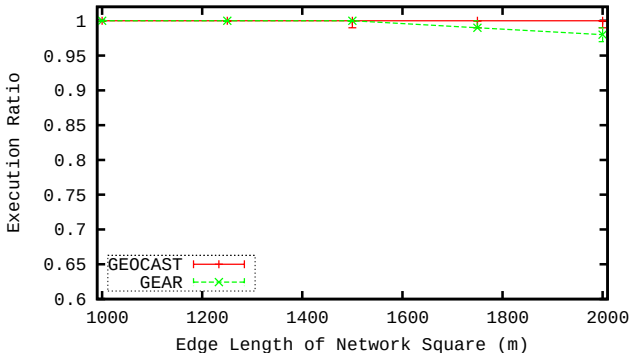
6.3.1. Scenario 1: Effects of network density

Scenario 1 evaluates the effect of varying network density by varying edge length of the (square) deployment area from $1000m$ to $2000m$. The number of nodes and gateways is fixed, 100 nodes and 3 gateways. Figure 5 shows the routing accuracy performance of this scenario.

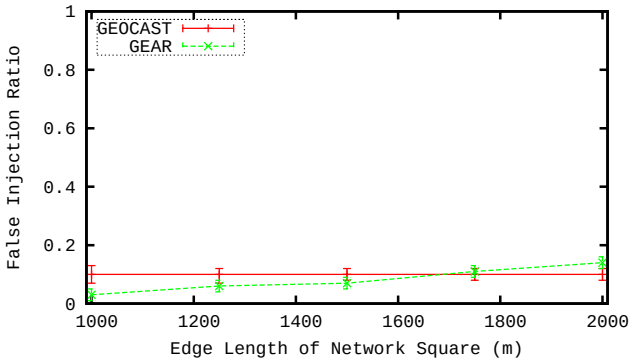
In Figure 5(a), we compare the ER of GEAR and GEOCAST for varying network density. Note that an execution ratio of 1 means that every node

inside the target region has received and executed the corresponding query packet. GEAR protocol, which has a higher redundancy than GEOCAST, has a lower execution ratio in sparse networks. This is mainly due to the unicast nature of the GEAR protocol. Its Greedy forwarding may not be able to find a path between a gateway and the target region when the network is sparse. On the other hand, even in sparse networks, in GEOCAST, the gateways can form their coverage areas including all sensors in the network. Therefore, nodes in the target region are accessible at least from one of the sinks in the network.

Some false injections exist even in static networks. Figure 5(b) compares the FIR of the two protocols. FIR of GEOCAST is constant because the protocol eliminates the unnecessary query injections from sinks both in dense and sparse networks. GEOCAST has always a small number of false injections due to the fact that the convex hull of a sink can overlap with the region of interest although no node from its partition is present in the target area. This effect is due to our choice of describing the coverage area with a convex hull. In dense networks, FIR of GEAR is less than FIR of GEOCAST. However, when the network is getting more sparse, FIR of GEAR increases. This is because all the sinks send the query packet, and the Greedy forwarding it uses for routing to the region of interest fails more often in sparse networks.

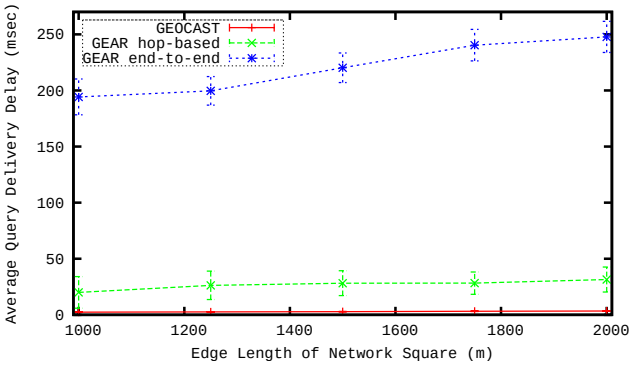


(a) Execution Ratio

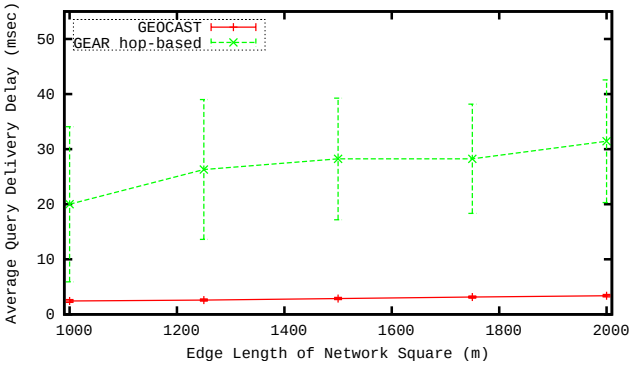


(b) False Injection Ratio

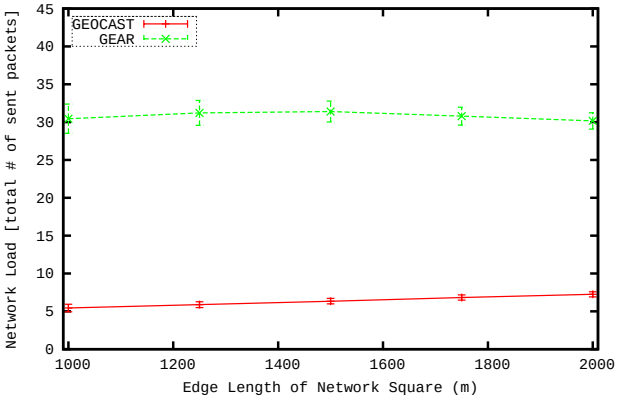
Figure 5: Scenario 1 – Routing accuracy performance for varying network density



(a) Average Query Delivery Delay (msec)



(b) Average Query Delivery Delay (msec)



(c) Network Load (number of sent packets)

Figure 6: Scenario 1 – Networking performance for varying network density

Figure 6 shows the networking performance of *Scenario 1*. Figure 6(a) shows the delay of GEOCAST, end-to-end delay of GEAR and hop-based delay of GEAR. Due to many agents called in GEAR (as explained in Section 6.2) the difference between GEAR end-to-end delay and hop-based delay is very big.

Figure 6(b) presents the hop-based delay of GEAR and end-to-end delay of GEOCAST. The end-to-end delay of GEOCAST is around 2 msec for a network of 1000x1000m^2 and 4 msec for a network of 2000x2000m^2. The delay of GEOCAST increases slightly with the increase of the network area. GEOCAST delay is smaller than GEAR delay because GEOCAST discards the queries of sinks which coverage area does not intersect with the region of interest. Only the sinks with overlapping coverage area with the target region inject their queries. These sinks are often closer to the region of interest than the other sinks. This keeps the average query delay low. In GEAR all the sinks inject the query, therefore the average query delivery delay gets higher. Another reason of this difference in delays of GEAR and GEOCAST is their different forwarding mechanisms. GEAR uses unicasting to reach next hop, and GEOCAST uses a special restricted broadcasting, which is quicker in the delivery of the message to the next node. Also, in the GEAR's delay graph, the 95% confidence range is very large, which shows a different response for different topology configurations.

Figure 6(c) shows the network load of both protocols. GEAR generates more network load than GEOCAST, since all of the sinks in GEAR try to reach the target region and inject a query packet into the network. The multiple forwarding paths also result in higher network load in GEAR. On

the other hand, GEOCAST checks the coverage areas of sinks and a sink sends a query packet only if its coverage area overlaps with the region of interest. Therefore, GEOCAST sends less query packets towards the target region. We can also say that GEOCAST is more energy efficient that GEAR because it sends less query packets, while having a very high query execution ratio.

6.3.2. Scenario 2: Effect of varying network size

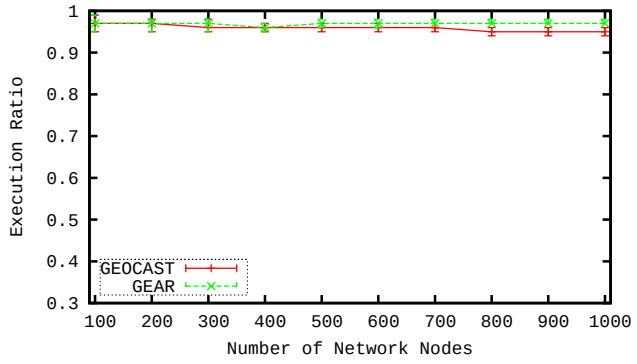
This scenario evaluates the effect of varying network size (from 100 to 1000 nodes), while the mean node density and the mean gateway load are kept constant. The average node density is set to 6 neighbors per node and the average gateway load —number of nodes connected to a gateway– is set to 33 nodes per gateway. In this scenario, the network area is increased with the number of nodes. For 100 nodes, the network size is 1800x1800m^2; for 1000 nodes, it is 5700x5700m^2. To keep the sink load constant we increase the number of sinks: 3 sinks in the network with 100 nodes, 30 sinks in the network with 1000 nodes. Figure 7 shows the routing accuracy performance of this scenario.

In Figure 7(a) we compare ER of GEAR and GEOCAST for varying number of network nodes. The ER of GEAR slightly outperforms the ER of GEOCAST when the number of network nodes increases. GEOCAST approach is less redundant than GEAR since in GEAR all sinks send the query towards the target region. Therefore, queries in GEAR reach to the target region via multiple paths, which results in higher ER.

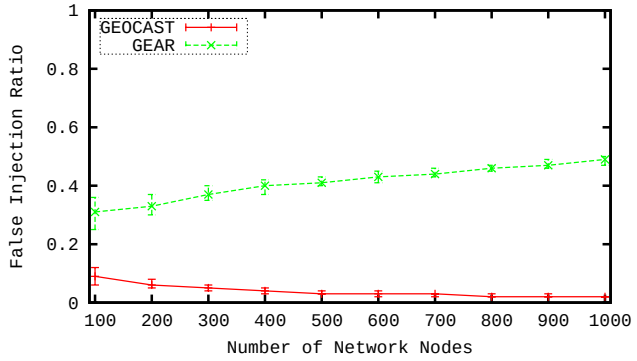
Looking at the FIR of the two protocols, Figure 7(b), we see that FIR of GEAR is getting higher when the number of network nodes increases. Since

we also increase the number of sink nodes, more sinks inject the query packets in GEAR. Therefore, the network with more sinks has a higher probability of query packets not reaching to the area of interest. On the other hand, the FIR of GEOCAST decreases as the number of sink nodes increases because more sinks mean more coverage areas that partition the network area, and the probability of the target region overlapping a coverage area decreases. In GEOCAST, gateways can decide to discard the query without inserting it in the WSN. Thus, GEOCAST can eliminate more unnecessary query injections in networks with more sinks and more nodes. FIR results show that the ratio of unnecessary injections of queries is very low in GEOCAST while GEAR has many unnecessary query insertions, which consume more energy and shorten the lifetime of the WSN.

Figure 8 shows the networking performance for varying number of nodes. As it can be seen in Figure 8(a), the query delivery delay of GEAR is increasing excessively with the number of network nodes, while the delay of GEOCAST is stable. the reasons are the same as for the previous scenario. GEOCAST only injects the queries of sinks which coverage areas overlap with the region of interest. These sinks are usually closer to the region of interest, therefore the paths from sinks to the target region are shorter. GEAR sends the queries of all sinks. When the network size increases, it results in more hops between the farthest sink and the target region, therefore longer delays for GEAR. Figure 8(b) presents the network load of the protocols. We notice that in both protocols, the network load increases with the increase of the number of nodes. However, the increase in network load of GEAR is very large since more sinks generate more query packets and all these query



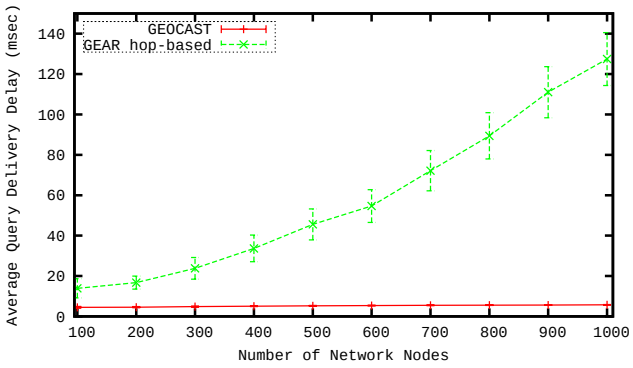
(a) Execution Ratio



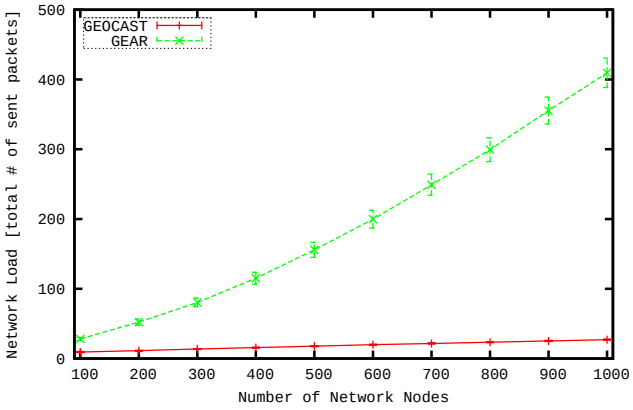
(b) False Injection Ratio

Figure 7: Scenario 2 – Routing accuracy performance for varying network size

packets try to reach the area of interest. GEOCAST, on the other hand, eliminates most of the unnecessary query injections so has a very small increase in the network load as the number of nodes and sinks increase. Summarizing results of Figure 7 and Figure 8, we can conclude that GEOCAST has a large energy saving even in big networks. Although GEAR uses multiple paths to forward queries towards the area of interest, the query execution ratio of GEOCAST is very close to the ER of GEAR.



(a) Average Query Delivery Delay (msec)



(b) Network Load (number of sent packets)

Figure 8: Scenario 2 – Networking performance for varying network size

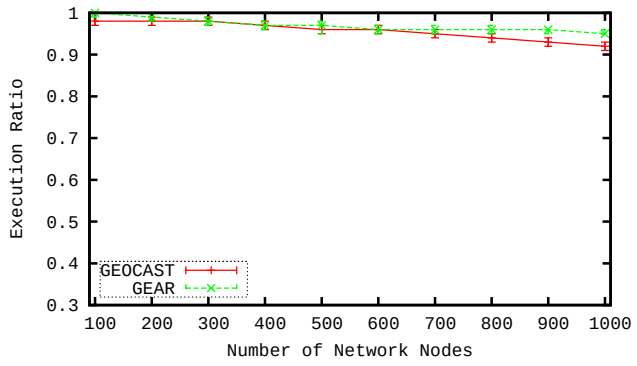
6.3.3. Scenario 3: Effects of gateway load

Scenario 3 evaluates the effect of the number of nodes associated per gateway in the network. This measures how the performance changes with the gateway load, which determines at the same time the average path length in the network. The gateway load is varied between an average of 11 nodes/gateway and 110 nodes/gateway. The node density is kept on average at 6 neighbor nodes and 9 gateways are used in the network setup. Both the deployment area and the number of nodes in the network vary in order to tune the gateway load.

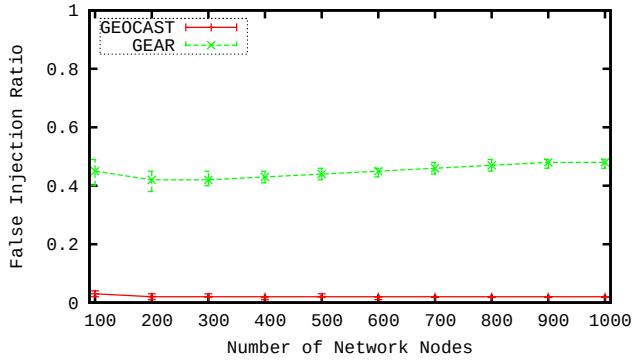
Figure 9 shows the routing accuracy performance of scenario 3 to see the effect of varying sink load. As shown in Figure 9(a), ER of GEAR and GEOCAST decrease when the load (the number of nodes) per gateway increases. ER of GEAR and ER of GEOCAST are very close to each other.

False injection ratios of both protocols remain more or less constant as the sink load increases, as shown in Figure 9(b). This is because the number of sinks is constant. GEAR has a higher FIR than GEOCAST. This is mainly due to the fact that GEOCAST eliminates unnecessary query injections by checking if the coverage area of a sink intersects with the target region. On the other hand, GEAR injects the queries from all sinks even though they may be far from the target region. The transmission of a query packet injected by the far away sinks may fail in the network due to the failure of pure Greedy forwarding at dead ends.

Figure 10(a) presents how the query delivery delays of both protocols are affected by the varying sink load. When we increase the sink load, the delay of GEOCAST slightly increases but the delay of GEAR increases very much.



(a) Execution Ratio

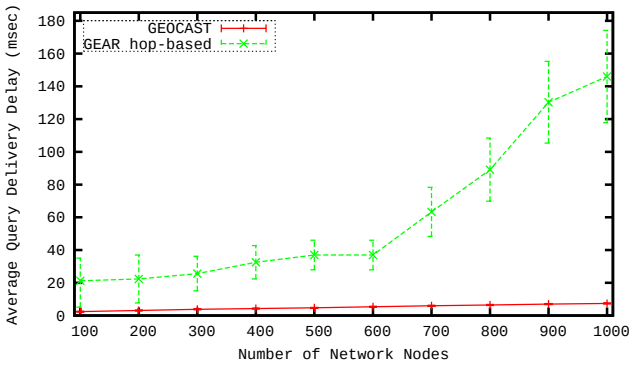


(b) False Injection Ratio

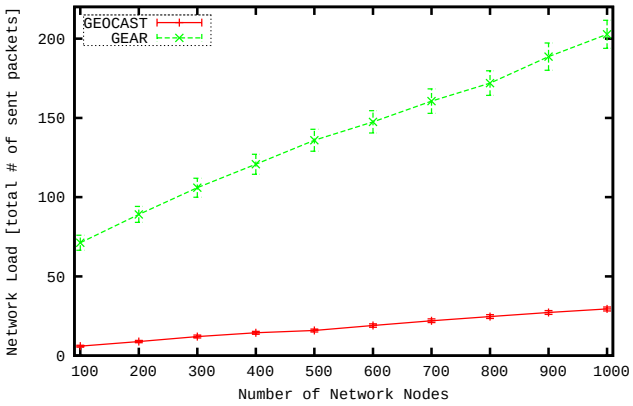
Figure 9: Scenario 3 – Routing accuracy performance for varying sink load

Having a fix number of sinks but increasing number of network nodes in a constant node density network, results in longer paths between sinks and target region. This effect is seen more clearly in the delay graph of GEAR because all sinks inject their queries into the network. Figure 10(b) shows the network load of the protocols when we increase the sink load. For the same reasons explained in previous scenarios, the network load of GEAR is very high because all the sinks inject the query in the network. In GEOCAST only few sinks inject the query into the WSN, usually those that are close

the region of interest. The longer paths are another reason for network load. This is more visible in GEAR because all the sinks send a query towards the target region, even those that are far away.



(a) Average query delivery delay (msec)



(b) Network Load (number of sent packets)

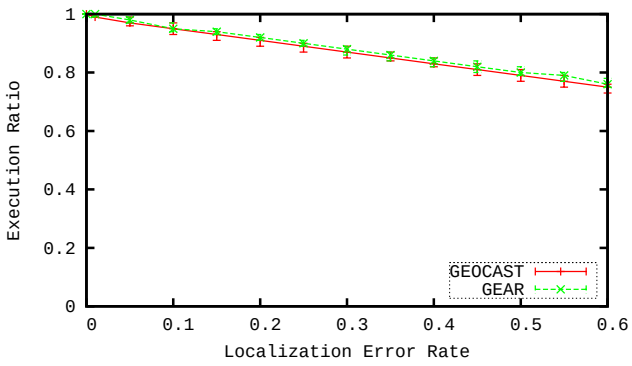
Figure 10: Scenario 3 – Networking performance for varying sink load

6.4. Scenario 4: Sensitivity to position estimate error

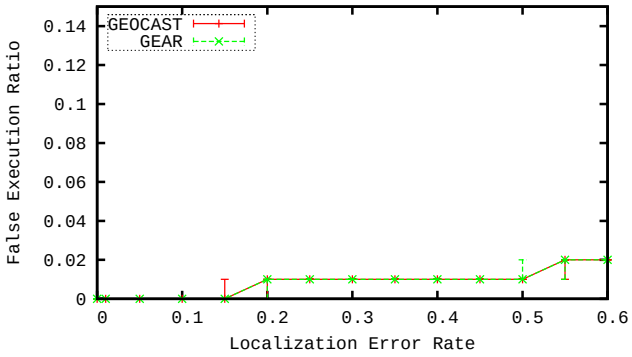
In this section we simulate positioning errors in static networks. In GEOCAST and GEAR is assumed that nodes are able to estimate their positions. However, most localization schemes introduce errors. As a consequence, in GEOCAST the service areas do not match the reality exactly and the execute decisions get less accurate, resulting in sub-optimal performance.

To investigate the sensitivity of both protocols to position estimate errors, we examine the impact of random location errors for every node in the network. The error in location was generated uniformly in the range [−error-rate * transmission-range, error-rate * transmission-range]. For example, if the error rate is 0.1 and the transmission range is $250m$, the randomized error is uniformly distributed in [-25, 25]. The following simulations are performed in a 1000x1000m^2 network having 100 nodes and 3 sinks. The error rate varies between 0 (i.e. no positioning errors) to 0.6, thus the randomized error is between $-150m$ and $150m$.

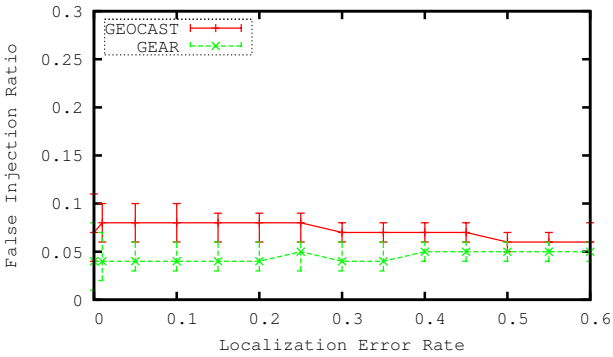
Figure 11 shows the routing accuracy performance of varying position estimate error rates. The results in Figure 11(a) show that ER decreases with increasing position estimation errors. However, with relatively big location error, both GEOCAST and GEAR protocols still achieve satisfying performance. For example, when we introduce random error in the range [-25, 25] for each node's x- and y-coordinates (i.e. error rate of 0.1), the simulation results show very small performance degradation: the ER is still 95% for both protocols.



(a) Execution Ratio

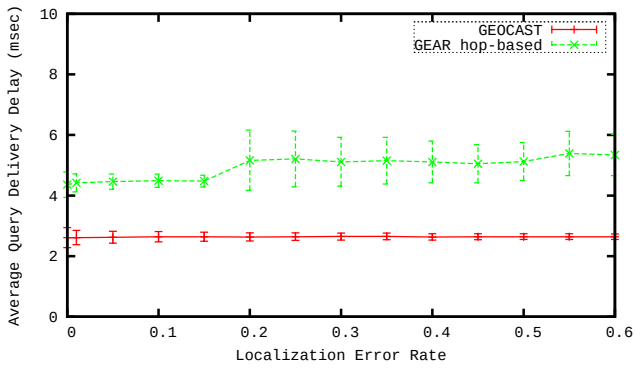


(b) False Execution Ratio

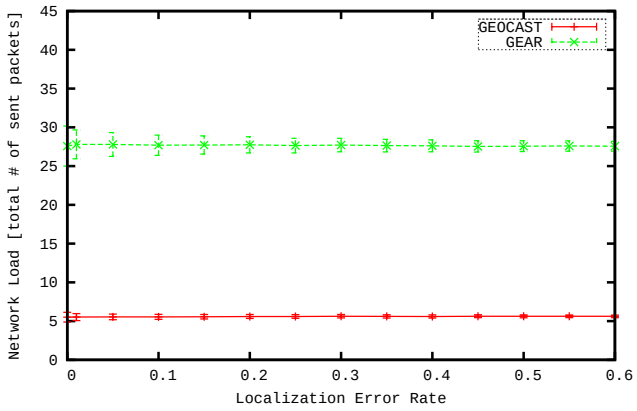


(c) False Injection Ratio

Figure 11: Scenario 4 – Routing accuracy performance for varying position estimate error



(a) Average query delivery delay (msec)



(b) Network Load (number of sent packets)

Figure 12: Scenario 4 – Networking performance for varying position estimate error

When we introduce a randomized error between [-150, 150] for each node's x- and y- coordinates (i.e. error rate of 0.6), the simulation results show around 25% performance degradation in ER. The sensitivity of both protocols to position estimate errors is very similar and ER of GEAR and GEOCAST are very close to each other. Since we introduce localization errors in the network, some false query executions exist in the network as shown in Figure 11(b). False execution ratios of the protocols are the same for a given

position estimate error rate. For an error rate of 0.15, i.e. around $37.5m$ error, FER of GEAR and GEOCAST is still zero. If we introduce an error rate of 0.2, 1% of the nodes outside the region of interest are executing the queries on average. FIR results shown in Figure 11(c) are not much affected from the increase of the positioning error. FIR of GEOCAST is slightly higher than FIR of GEAR. However, when we increase the localization error rate, the FIRs of both protocols are more or less the same. Coverage region based approach of GEOCAST allows better hiding of inaccuracy of node positions for higher localization errors. On the other hand, using individual inaccurate positions for geographical routing in GEAR results in more failures of query forwarding for higher error rates.

Figure 12 shows the networking performance of varying position estimate error. In Figure 12(a), query delivery delays of GEAR and GEOCAST are shown. The query delivery delay of GEAR increases progressively when the localization error rate increases. On the other hand, GEOCAST has a constant delay with increasing localization error rate.

GEOCAST is based on convex hulls of sinks' coverage areas and even if we have localization errors in the nodes, the resulting convex hulls are the approximation of the correct convex hulls. Since we have a hierarchical structure in the definition of coverage areas in GEOCAST, a node has an 'overview' about the region between its position and the area of interest. This 'overview' is affected by localization errors, but is still a good approximation of the correct coverage area. Therefore, the path between a sink and the area of interest is also an approximation of the correct path between the sink and the area of interest. As a results, the delay of GEOCAST is not affected by

increasing localization errors.

However, routing strategies such as GEAR, which carry the packet geographically closer to the destination in each hop, can result in different paths with different localization errors. Since a node in GEAR only has a local knowledge about its surrounding nodes (i.e. one hop neighbors' positions) and it forwards the message based on this local knowlegde, the path between sink and the target region can be longer when nodes have position estimate errors. In Figure 12(a), we observe higher delays with higher localization errors in GEAR, and we expect that to be caused by longer paths between the sinks and the nodes in the target region.

The network loads of both protocols are constant for varying error rates. Even if we have position estimate errors in the network, both GEAR and GEOCAST try to forward queries towards the area of interest. Therefore, the network load is constant when we increase the error rate. The number of packets the protocol sends when there is no positioning error in the network is the same with the number of packets sent when there are positioning errors. Although network load of GEAR and GEOCAST do not change with the increase of error rate, the ER of both protocols is affected, because the protocols cannot deliver the query packets to the correct locations due to positioning errors.

6.5. Mobile networks

The effects of mobility on both protocols are evaluated in scenarios 5 and 6. The mobile network simulations are performed in a 2500x2500m^2 deployment area with 200 sensor nodes and 6 gateways. Simulations of both mobile scenarios show that the False Execution Ratio of queries in mobile

networks is zero for both GEAR and GEOCAST protocols, therefore not shown in the graphs. This is due to the fact that when a mobile node that was in the target region before starting to move, receives a query, it first checks it current position and if it is not inside the area of interest anymore, it does not execute the query.

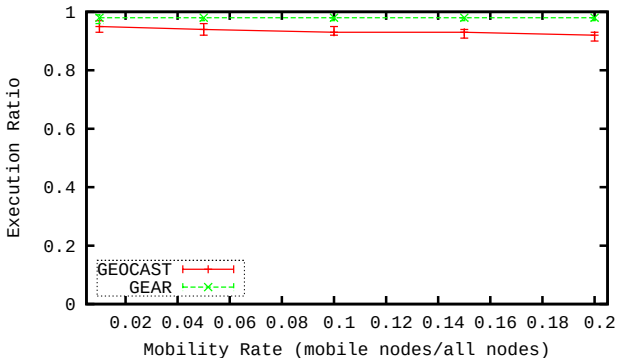
6.6. Scenario 5: Effect of the number of mobile nodes

This scenario evaluates the effects of the different number of mobile nodes in a network: 1% to 20% of the network nodes are moving. The average speed of the mobile nodes is $5m/s$. Figure 13 shows the routing accuracy performance of this scenario.

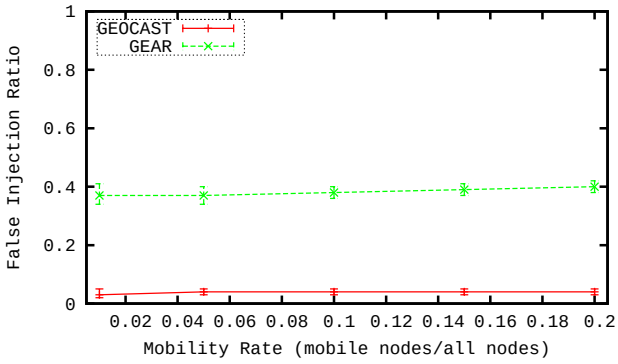
Figure 13(a) shows the ER results for GEAR and GEOCAST. GEAR slightly outperforms GEOCAST when we increase the mobility rate. This is mainly due to the fact that we used tree-based shortest path routing to connect nodes to sinks in GEOCAST simulations. As tree-based approaches require frequent reconfigurations in mobile sensor networks, they may have worse performance in mobile sensor networks.

Figure 13(b) presents the FIR results of GEAR and GEOCAST. As we already observe in the previous simulations, the FIR of GEAR is higher than the FIR of GEOCAST. When we increase the mobility rate in the network, the FIR of GEAR slightly increases. GEOCAST also has a slight increase in FIR when the mobility rate is increased. This means that although convex hulls of sensors may be affected by mobile nodes, the coverage areas of gateways are not much affected. Only the movement of nodes that are close to the boundary of a gateway's convex hull may change the coverage area of this gateway when they pass this boundary.

Figure 14 shows the networking performance of this scenario. In Figure 14(a) we show the query delivery delays of both protocols. The delays are not affected much by the increasing number of mobile nodes in the network. The delay is still much higher in GEAR than in GEOCAST. The network loads of GEAR and GEOCAST are shown in Figure 14(b). As we already see in the previous simulations, the network load of GEAR is higher than the network load of GEOCAST, also in this mobile scenario. The network load of GEAR is slightly decreasing with the increase of the number of mobile nodes. This is to be seen together with the ER of GEAR. The probability of a neighborhood change is getting higher with the increase of mobility rate. A query packet can get stuck at a node that does not have anymore a neighbor closer to the region of interest. Therefore, the number of sent query packets decreases in GEAR.

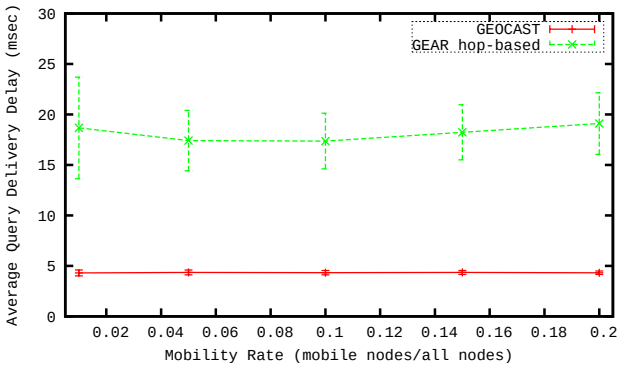


(a) Execution Ratio

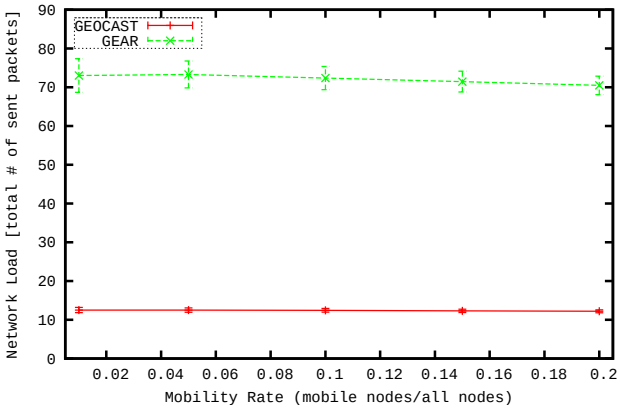


(b) False Injection Ratio

Figure 13: Scenario 5 – Routing accuracy performance for varying mobility rate



(a) Average Query Delivery Delay (msec)



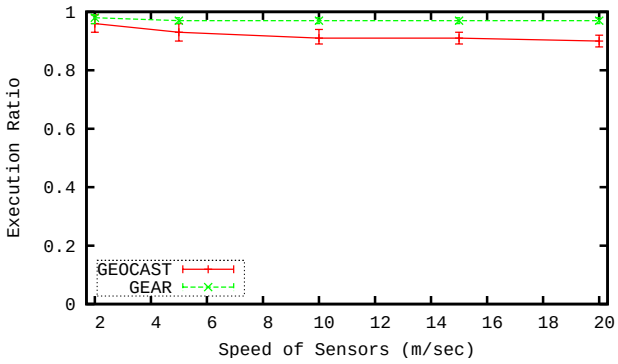
(b) Network Load (number of sent packets)

Figure 14: Scenario 5 – Networking performance for varying mobility rate

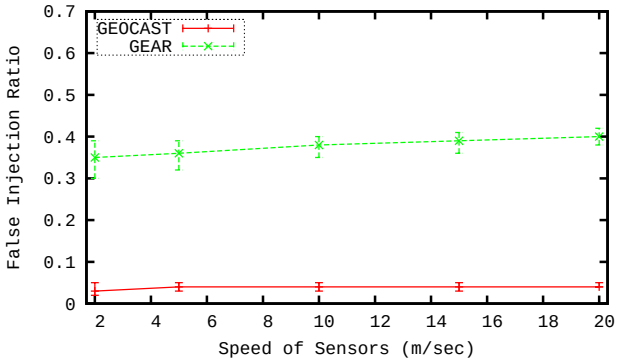
6.7. Scenario 6: Effect of speed of mobile nodes

The speed of mobile nodes is varied in this scenario: 5% of the network nodes are moving with a speed varying between $2m/s$ and $20m/s$. Figure 15 shows the routing accuracy performance. The results in Figure 15(a) and Figure 15(b) show the same behavior as the simulation results of scenario 5. Higher mobility speed requires more frequent reconfigurations in mobile sensor networks in GEOCAST. Therefore, ER of GEOCAST is less than ER of GEAR due to the tree-based structure used by GEOCAST, but the difference is very small. FIR results of GEAR and GEOCAST in scenario 6 are also very similar to the results in scenario 5.

Figure 16 shows the networking performance of scenario 6. Figure 16(a) presents the query delivery delays of both protocols when we increase the speed of mobile sensors. The query delay of both protocols is affected very little by the increase on nodes speed. GEOCAST is outperforming GEAR considerably. Figure 16(b) shows the effect of nodes speed on the network load. GEOCAST network load is unaffected by the change in mobile nodes speed. It is again much lower than the network load of GEAR.

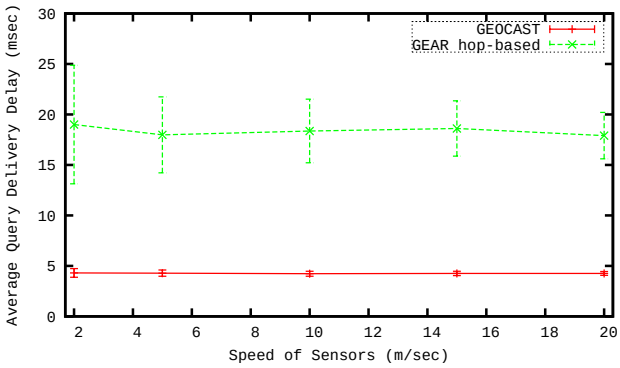


(a) Execution Ratio

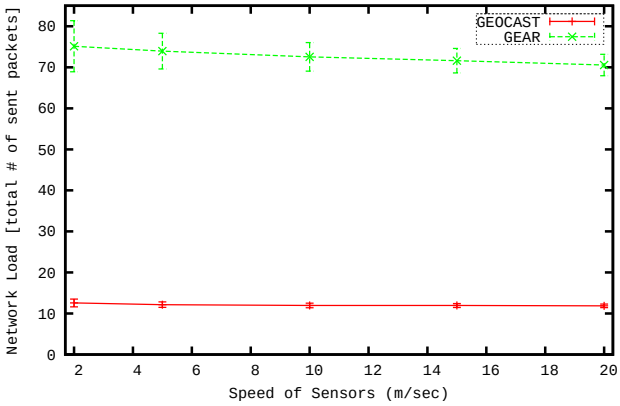


(b) False Injection Ratio

Figure 15: Scenario 6 – Routing accuracy performance for varying mobility speed



(a) Average Query Delivery Delay (msec)



(b) Network Load (number of sent packets)

Figure 16: Scenario 6 – Networking performance for varying mobility speed

7. Conclusions

In this paper we proposed the GEOCAST protocol for multi-sink WSNs, which performs a geographical routing of queries to a region of interest. WSN nodes are partitioned between different gateways and organized in trees, each tree rooted at a gateway. The GEOCAST protocol creates a hierarchical structure over a routing tree that consists of nested convex hulls. The convex hull of the root is the coverage area of that gateway. The convex hull of any other node in the tree encapsulates the area served by this node and all its dependents in the rooting tree, forming a local coverage area. The (local) coverage areas are used for the routing of queries that are valid for a geographical region. Gateways check their coverage area against the region of interest of the query. The query is injected from a gateway to the WSN only if its coverage area overlaps with the region of interest. A node further down in the tree executes the query if it is within the region of interest, or it forwards it to its descendants if its local coverage area overlaps with the region of interest. GEOCAST has been evaluated in terms of computational complexity and memory requirements.

We compared our protocol with GEAR, which is one of the best performing geocasting protocols. The comparison was done for both static and mobile scenarios (i.e. WSN contain mobile nodes) in terms of routing accuracy and on networking. Three metrics were employed to test the routing accuracy: execution ratio, false execution ration, and false injection ratio. Two more metrics are used for the networking performance: average query delay and network load. ... something on the results.

Both GEAR and GEOCAST assume that nodes have an estimate of their

position. We tested and compared the two protocols for the effect of positioning error. ... more on results GEOCAST experiences a slightly higher false injection ratio. In mobile scenarios, GEOCAST experiences a lower execution ratio than GEAR, but again it needs less messages to deliver messages and GEOCAST has a smaller false injection ratio. For applications that require mobility in the wireless sensor network, the trade-off should be made if a potential lower energy-consumption of GEOCAST is more preferable than the higher routing accuracy of GEAR.

Acknowledgements

We gratefully thank Yan Yu from the University of California, Los Angeles, for the GEAR implementation in NS2. This work has been partly sponsored by the FREE project (funded by the Dutch innovation programme PointOne organized by AgentschapNL).

References

[1] S. Chatterjea, L. van Hoesel, and P. Havinga. AI-LMAC: An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks. In Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference, pages 381–388, IEEE Computer Society Press, 2004.

[2] P. Levis, N. Patel, D. Culler, and S. Shenker Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks In: Proceedings of the First Symposium on Networked Systems Design and Implementation March 29–31, 2004.

[3] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In 2nd ACM conf. on Embedded Networked Sensor Systems, pages 81–94, November 2004.

[4] N. Reijers and K. Langendoen. Efficient Code Distribution in Wireless Sensor Networks. In Proc. 2nd ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA), San Diego, CA, September 2003.

[5] S.R. Madden. The Design and Evaluation of a Query Processing Architecture for Sensor Networks. PhD thesis, University Of California, Berkeley, 2003.

[6] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. In ACM Transactions on Database Systems (TODS), Vol. 30–1 March 2005, pages 122–173, ISSN 0362-5915.

[7] T.van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In: 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003), pages 171–180, November 2003.

[8] M. Zorzi and R.R. Rao. Geographic random forwarding (GERAF) for ad hoc and sensor networks: Multihop performance. In: IEEE Transactions on Mobile Computing, Vol. 2, No. 4, pages 337–348, 2003.

[9] E. Kranakis, H. Singh and J. Urrutia. Compass Routing on Geometric Networks, In: Proceedings of 11th Canadian Conference on Computational Geometry (CCCG). Vancouver, August 1999, pp. 5154.

[10] Prosenjit Bose, Pat Morin, Ivan Stojmenovi and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In: Wireless Networks, 7(6):609–616, 2001.

[11] F. Kuhn, R. Wattenhofer and A. Zollinger. Asymptotically Optimal Geometric Mobile Ad-hoc Routing, In: Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Atlanta, Georgia, USA, September 2002, pp. 24–33.

[12] B. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: Proceedings of 6th Annual Int. Conf. on Mobile Computing and Networking (MobiCom), Boston, Massachusetts, USA, pages 243–254, 2000.

[13] F. Kuhn, R. Wattenhofer and A. Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-hoc Routing. In: Proceedings of 4th ACM Int. Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), Annapolis, Maryland, pp.267–278, 2003.

[14] F. Kuhn, R. Wattenhofer, Y. Zhang and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice, In: Proceedings of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC), 2003.

[15] Y. Yu, R. Govindan and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks.

Technical Report UCLA/CSD-TR-01-0023, University of Southern California, 2001.

[16] Y. Ko and N. Vaidya. Anycasting-based Protocol for Geocast service in Mobile Ad Hoc Networks. In: Computer Networks Journal, 2003.

[17] V. Park and M. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In: Proceedings of the 6th IEEE INFOCOM, Kobe, Japan, 1997.

[18] A. Coman, M.A. Nascimento and J. Sander. A framework for spatio-temporal query processing over wireless sensor networks, In: Proceedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004, Toronto, Canada, August 30-30, 2004.

[19] I. Stojmenovic. Geocasting with guaranteed delivery in sensor networks. In: IEEE Wireless Communications Magazine, 11(6):2937, 2004.

[20] K. Seada and A. Helmy. Efficient and robust geocasting protocols for sensor networks. In: Elsevier Computer Communications, 29(2):151-161, 2006.

[21] C. Maihöfer. A survey of geocast routing protocols. In: IEEE Communications Surveys and Tutorials, Vol. 6, No. 2, pages 32–42, April 2004.

[22] Y.B. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In: Proceedings of 4th ACM/IEEE Int. Conf. Mobile Computing (MobiCom), Dallas, USA, 1998.

[23] I. Stojmenovic, A. P. Ruhil and D.K. Lobiyal. Voronoi diagram and convex hull based geocasting and routing in wireless networks. In: IEEE Symposium on Computers and Communications (ISCC), Antalya, Turkey, July 2003, pp.51–56.

[24] W.-H. Liao, Y.-C. Tseng, K.-L. Lo, J.-P. Sheu, GeoGRID: a geocasting protocol for mobile ad hoc networks based on GRID. In: Journal of Internet Technol. 1(2):23-32, 2000.

[25] A. B. Bomgni, J. F. Myoupo. An Energy-Efficient Clique-Based Geocast Algorithm for Dense Sensor Networks. Communications and Network, 02(02):125-133, 2010.

[26] A. Baggio and K. Langendoen. Monte-Carlo Localization for Mobile Wireless Sensor Networks. Elsevier's Ad Hoc Networks Journal, vol. 6, no. 5, July 2008.

[27] J. Hightower and G. Borriello. SPOTON: An indoor 3D location sensing technology based on RF signal strength. Technical Report University of Washington, February 2000.

[28] D. Niculescu and B. Nath. Ad hoc positioning system (APS). IEEE Global Telecommunications Conference (GLOBECOM '01), pp. (5)2926–2931, 2001.

[29] T. He, C. Huang, B.M. Blum, J.A. Stankovic, T. Abdelza-her. Range-free localization schemes for large scale sensor networks. In MobiCom 2003, San Diego, CA, USA, September 2003.

[30] B. Dil, S.O. Dulman, and P.J.M. Havinga. Range-Based Localization in Mobile Sensor Networks. In: Proceedings of Third European Workshop on Wireless Sensor Networks, 13-15 Feb 2006, Zurich, Switzerland. pp. 164–179. Lecture notes in computer science 3868. Springer Verlag, ISBN 3-540-32158-6, 2006.

[31] C. Fischer, K. Muthukrishnan, M. Hazas, and H. Gellersen. Ultrasound-Aided Pedestrian Dead Reckoning for Indoor Navigation. In: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments, Co-located MOBICOM 2008, 15–19 September 2008, San Francisco, USA. pp. 31–36.

[32] Bang Wang, Wei Wang, Vikram Srinivasan, and Kee Chaing Chua. Information Coverage for Wireless Sensor Networks. In: IEEE Communications Letters, Vol. 9–11, pages 967–969, November 2005, ISSN 1089-7798.

[33] K. Römer and F. Mattern. The Design Space Wireless Sensor Networks. In: IEEE Wireless Communication Magazine, Vol. 11, No. 6, pages 54–61, 2004, ISSN 1536-1284.

[34] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. In: IEEE Transactions on Information Theory, IT-29(4):551–559, 1983.

[35] N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E.P. Mücke and C. Varela. Alpha Shapes: Definition and Software. In: Proceedings of the

1st International Computational Geometry Software Workshop, pages 63–66, 1995.

[36] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Elsevier Computer Networks, Vol. 38, No. 4, pages 393–422, 2002.

[37] H. Zimmermann. OSI Reference Model – The ISO model of architecture for open systems interconnection. In: IEEE Transactions on Communications, Vol. 28, No. 4, April 1980.

[38] Y. Bejerano, S.J. Han and A. Kumar. Efficient Load-Balancing Routing for Wireless Mesh Networks. In: Computer Networks, Vol. 51, No. 10, pages 2450–2466, 2007, ISSN 1389-1286.

[39] Y. Zhang and Q. Huang. A Learning-based Adaptive Routing Tree for Wireless Sensor Networks. In: Journal of Communications, Vol. 1, No. 2, pages 12–21. Academy Publisher, 2006, ISSN 1796-2021.

[40] A. Erman-Tüysüz, T. Mutter, L.F.W. van Hoesel, and P.J.M. Havinga. A Cross-Layered Communication Protocol for Load Balancing in Large Scale Multi-sink Wireless Sensor Networks. In: Proceedings of the Nineth International Symposium on Autonomous Decentralized Systems (ISADS), 23-25 March 2009, Athens, Greece. IEEE Computer Society.

[41] A. Erman-Tüysüz, L.F.W. van Hoesel, P.J.M. Havinga, and J. Wu. Enabling Mobility in Heterogeneous Wireless Sensor Networks Cooperating with UAVs for Mission-Critical Management. In: IEEE Wireless Communications, Vol. 15, No. 6, pages 38–46, 2008, ISSN 1536-1284.

[42] M.T. de Berg et al. Computational geometry: algorithms and applications. Third edition, Springer, Berlin, 2008, ISBN 978-3-540-77973-5.

[43] E. W. Weisstein. Line-Line Intersection. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Line-LineIntersection.html. Last accessed: November 2009.

[44] E. Haines. Point in Polygon Strategies. In: Graphics Gems IV, editor P. Heckbert, Academic Press, pages 24–46, 1994, ISBN 0-12-336155-9.

[45] S. Schirra. How Reliable Are Practical Point-in-Polygon Strategies? In: Lecture Notes in Computer Science, Algorithms - ESA 2008, Springer Berlin / Heidelberg, Vol. 5193, pages 744–755, 2008, ISBN 978-3-540-87743-1.

[46] B. Chazelle and D. Dobkin. Decomposing a polygon into its convex parts. In: Proceedings of the 11th annual ACM Symposium on Theory of Computing, Atlanta, United States, pages 38–48, ACM, 1979.

[47] D. H. Greene. The decomposition of polygons into convex parts. In Franco P. Preparata, editor, Computational Geometry, volume 1 of Adv. Comput. Res., pages 235-259. JAI Press, Greenwich, Conn., 1983.

[48] L.F.W. van Hoesel. Sensors on speaking terms: Schedule-based medium access control protocols for wireless sensor networks. PhD thesis, Univ. of Twente, 2007, ISBN 978-90-365-2497-1.

[49] R.B. Muhammad. Incremental Convex Hull. http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/ConvexHull/incrementCH.htm. Last accessed: June 2010.

[50] J. O'Rourke. Computational Geometry in C. Cambridge University Press, 1998, ISBN 0-521-64010-5.

[51] C.W. Huang and T.Y. Shih. On the complexity of point-in-polygon algorithms. In: Computers and Geosciences, Vol. 23, No. 1, pages 109–118, Elsevier, February 1997, ISSN 0098-3004.

[52] B. Chazelle and D. Dobkin. Detection is easier than computation. In: Proceedings of the 12th annual ACM Symposium on Theory of Computing, Los Angeles, United States, pages 146–153, ACM, 1980.

[53] L.F.W. van Hoesel, S.O. Dulman, P.J.M. Havinga, and H.J. Kip. Design of a low-power testbed for Wireless Sensor Networks and verification. Technical Report TR-CTIT-03-45, Centre for Telematics and Information Technology, University of Twente, Enschede, 2003, ISSN 1381-3625.

[54] The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/. Last accessed: June 2010.

[55] J. Heidemann, F. Silva, Y. Yu, D. Estrin and P. Haldar. Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Networks. In Technical Report of USC/ISI, ISI-TR-556, 2002.