Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2010, Article ID 679205, 15 pages
doi:10.1155/2010/679205

Research Article

Position-Based Aggregator Node Election in Wireless Sensor Networks

Levente Buttyán[1] and Péter Schaffer[2]

[1] *Laboratory of Cryptography and Systems Security (CrySyS), Budapest University of Technology and Economics (BME), 1117 Budapest, Hungary*
[2] *Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg (UL), 1359 Kirchberg, Luxembourg*

Correspondence should be addressed to Péter Schaffer, peter.schaffer@uni.lu

Received 25 September 2008; Accepted 13 August 2009

We introduce PANEL a position-based aggregator node election protocol for wireless sensor networks. The novelty of PANEL with respect to other aggregator node election protocols is that it supports asynchronous sensor network applications where the sensor readings are fetched by the base stations after some delay. In particular, the motivation for the design of PANEL was to support reliable and persistent data storage applications, such as TinyPEDS; see the study by Girao et al. (2007). PANEL ensures load balancing, and it supports intra and intercluster routing allowing sensor-to-aggregator, aggregator-to-aggregator, base station-to-aggregator, and aggregator to-base station communications. We also compare PANEL with HEED; see the study by Younis and Fahmy (2004) in the simulation environment provided by TOSSIM, and show that, on one hand, PANEL creates more cohesive clusters than HEED, and, on the other hand, that PANEL is more energy efficient than HEED.

1. Introduction

Wireless sensor networks consist of a multitude of tiny sensor nodes capable for wireless communications and a few powerful base stations. The sensor nodes usually perform some monitoring task (e.g., measure various environmental parameters). The base stations collect sensor readings and forward them for further processing to a service center.

Based on how the sensor readings reach the base stations, we can distinguish *synchronous* and *asynchronous* sensor networks. In the synchronous case, the sensor readings are sent to the base stations in realtime using multihop wireless communications, where the sensor nodes cooperatively forward data packets on behalf of other sensor nodes towards the base stations. In the asynchronous case, the sensor readings are fetched by the base stations after some delay (e.g., once every day or week). In this case, the base stations are often mobile, and they physically approach the sensors in order to fetch their data through a single wireless hop. Examples of synchronous sensor network applications include forest fire alarm systems and building automation systems where realtime operation is indispensable. Examples of asynchronous applications include habitat monitoring systems and agricultural applications such as vineyard monitoring where realtime operation is not an issue.

As sensor nodes are often severely resource constrained, various techniques have been proposed to ensure the efficient operation of sensor networks. One of these techniques is called *aggregation* or *in-network processing*. The idea is that, instead of forwarding (in case of synchronous applications) or storing (in case of asynchronous applications) raw sensor readings, data can be first processed, combined, and compressed by some distinguished sensor nodes, called *aggregators*.

While aggregation increases the overall efficiency of the sensor network, the aggregator nodes themselves use more resources than the regular sensor nodes. For this reason, it is desirable to change the aggregators from time to time, and thereby, to better balance the load on the sensor nodes. For this purpose, aggregator node election protocols can be used in the sensor network that allow dynamic reassignment of the aggregator role.

In this paper, which is an enhanced version of our previously published conference paper in [1], we introduce PANEL: a position-based aggregator node election protocol for wireless sensor networks. As its name indicates, PANEL uses the geographical position information of the nodes to determine which of them should be the aggregators. Like other aggregator node election protocols, PANEL also ensures load balancing in the sense that each node is an elected aggregator nearly equally frequently. The salient feature of PANEL that makes it novel and different from other aggregator node election protocols is that besides synchronous applications, PANEL also supports asynchronous applications.

In particular, the motivation for the design of PANEL was to support TinyPEDS (Tiny Persistent Encrypted Data Storage) [2], and other similar asynchronous sensor network applications. In TinyPEDS, aggregator nodes collect and aggregate sensor readings from the clusters that they are responsible for, and then persistently store the aggregated values (in an encrypted form). In addition, in order to increase reliability, the aggregators replicate their stored data at the aggregators of some selected backup clusters. These backup aggregators (i.e., the aggregators in the backup clusters) must be chosen in such a way that they are farther away from the primary aggregator than a certain distance called the *disaster radius*. The rationale is that if there is a disaster in which the primary aggregator is destroyed, its data is still available at and can be retrieved from the backup aggregators. Being a position-based protocol, PANEL supports TinyPEDS and applications alike by providing assurances regarding the distance between the elected aggregator nodes.

The organization of the paper is the following. In Section 2, we report on the related work. In Section 3, we introduce the general assumptions that we based the design of PANEL upon. In Section 4, we describe the operation of PANEL. In Section 5, we present our simulation-based comparison of the performance of PANEL with that of HEED [3]: an aggregator node election protocol well known from the literature. In Section 6, we discuss some possible extensions of PANEL. And finally, in Section 7, we conclude the paper.

2. Related Work

Dividing the sensor network into clusters and using in-network aggregation is an effective way to treat the enormous amount of information produced by the sensor nodes. Several papers have been published in this research area for sensor networks; we list the related ones below.

There are some papers that give an overview of in-network aggregation and cluster formation solutions. One of these papers is that in [4], which discusses the main issues of clustering in sensor networks and concludes that clustering is a useful tool for topology management and for in-network data aggregation. In [5], the authors detail the different data aggregation techniques and highlight the tradeoffs between energy efficiency, data accuracy, and latency. In [6], one can read about a comparison of homogeneous (i.e., all of the nodes have same hardware capability) and heterogeneous (i.e., nodes have different hardware capabilities) sensor networks from clustering point of view, while in [7], the authors investigate the schemes of single-hop and multihop communications and their impact on clustering.

The papers that propose solutions for clustering can be classified based on their primary aim (however, most of these papers have multiple aims). The largest group according to this classification consists of papers that aim at lifetime maximization of the sensor network. This group of papers can be further divided based on the method that they use for the clustering: it can be either probabilistic or deterministic. In case of probabilistic solutions, the cluster heads are elected based on some randomness, while in case of deterministic solutions, some iterative or centralized strategies are deployed.

The most known probabilistic cluster formation algorithm is the LEACH protocol [8, 9]. In LEACH, the clustering goes as follows: in every round, each sensor node picks a random number, and if this random number is smaller than a threshold, the node becomes a cluster head. Next, it advertises itself with constant energy via radio communication and waits for cluster members. The cluster members are the noncluster head nodes, and each of them joins that cluster head's cluster whose advertisement was received with the highest energy (in general, this is the nearest cluster head's cluster). The properties of LEACH are that it flatly balances the energy consumption of the network; however, it uses only one-hop communication; the remaining energy of the nodes is not a parameter by the election (but it should be because of the increased energy needs of a cluster head node); and the protocol requires every node to be able to reach the base station in one hop, which is not generally true in sensor networks.

Other related probabilistic cluster formation solutions can be found in [10–15]. In [10, 11], one can read about a data gathering scheme, called PEGASIS, that forms a chain of nodes and elects a leader node for energy-efficient collection of the sensors' measurements. In [12], the authors propose a technique to build k-hop clusters, that is, where the cluster members are at most k-hops away from the cluster head. In [13], a LEACH-like idea is exploited considering the residual energy of the nodes as well. In [14, 15], the problem of unequal-energy dissipation is considered in case of equally sized clusters. Therefore, in these latter papers, the authors propose to form unequal size clusters: smaller ones close to the base station and larger ones further from it, as the cluster heads of closer-lying clusters have more load due to the message forwarding task for further-lying clusters.

The most important probabilistic solution for our purposes is HEED [3], which can also be considered as the generalization of [8] and [13]. In HEED, the cluster formation algorithm is more sophisticated; it elects the cluster heads based on their remaining energy and on a secondary parameter that can control the cluster density, the load balancing, and the amount of intracluster communication. The HEED protocol seems to be a good tradeoff between termination speed and cluster head distribution by allowing some communication between neighboring nodes. The simulation results of HEED show that it outperforms LEACH in terms of network lifetime and in ratio of energy dissipated for clustering.

The list of papers that aim at network lifetime maximization using a deterministic aggregator node election method is quite extensive as well. In [16–18], one can find solutions for the mentioned problem assuming that the sensor network is heterogeneous; that is, there are less-energy-constrained gateway nodes among the usual constrained sensor nodes that can help in cluster formation and in the routing of sensor messages. The authors of [19, 20] approach the clustering problem from data gathering point of view, and propose a centralized solution for near-optimal scheduling of the message sending. In [21], the problem of lifetime maximization is handled by balancing the load of cluster heads and by minimizing the total distance between sensor nodes and cluster heads. The study in [22] tackles the same problem using fuzzy logic with the variables of energy, node concentration, and node centrality with respect to the entire cluster.

There are some papers that do not primarily aim at network lifetime prolongation, but at quick cluster formation. In [23], the authors consider event-driven sensor networks with high degree of spatial-temporal correlation. The main focus of the paper is on cross-layered design of localized algorithms for performing quick data aggregation and quick hierarchy formation allowing prompt response to queries. In [24], one can read about a fast clustering algorithm suitable for large-scale sensor networks by its property of locality, scalability, and self-healing in case of node failures and newly deployed nodes.

The following group of papers collects those works that aim at enhanced network management. For example, the objective of [25] is to show that one can efficiently compute an asymptotically optimal clustering, even when collision resistant packet forwarding is not ensured. The authors of [26] propose a hierarchical clustering approach, where the cluster heads are clustered again at the higher layer of the hierarchy. Here, the clustering problem is defined in a graph theoretic framework, and a distributed solution is presented that results in size-bounded clusters. The ACE algorithm [27] aims at forming minimally overlapping clusters with the help of cluster migration. The algorithm ends in constant time regardless of the size of the network and uses only local communications between nodes.

Security is rarely considered in the cluster formation or aggregator node election problem. An exception is the study in [28], where the authors deal with the issue of nonmanipulable aggregator node election. When an attacker node is able to manipulate the aggregator election process, it is able to influence the operation of the network, for example, by electing itself as aggregator and maliciously filtering the sensors' measurements. Three independent countermeasures are proposed against such an attacker in [28], with all of them based on distributed random number generation. Another exception is the paper in [29], where the author proposed a technique for resilient cluster formation, which consists of a neighbor validation module based on wormhole detection, a priority-based selection of the cluster head nodes, and a centralized detection module that aims at detecting the nodes that have an abnormally large number of neighbors (as these are most likely compromised).

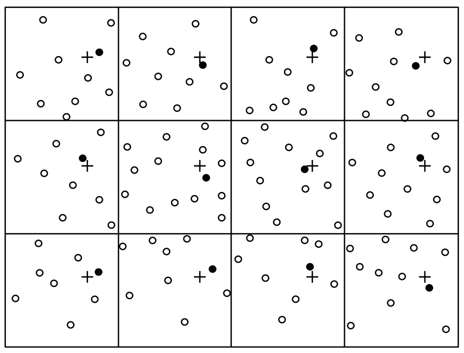
The papers listed above are all related to the aggregator node election problem assuming clustering. However, none of the above methods are able to guarantee a minimum distance between certain aggregators. However, in our motivating application area (i.e., reliable and persistent distributed data storage), backup aggregators must be chosen in such a way that they reside farther away from the primary aggregator than a certain disaster range. In [30], the authors detail an approach for aggregator node election that is able to guarantee this minimum distance; however, the proposed solution is centralized, and thus, its applicability is limited. On the contrary, PANEL can guarantee a minimum distance between aggregators in a distributed manner, because in PANEL the aggregator nodes reside within fixed-size clusters and are elected locally without the need of a central controller. For instance, the minimum distance between two aggregators belonging to nonneighboring clusters is dx, where x is the number of clusters between the two aggregators and d is the physical size of the cluster.

3. General Assumptions

One of the main assumptions that PANEL relies on is that the sensor nodes are static and they are aware of their geographical position. This is obtained either by means of GPS or by using any of the numerous node positioning algorithms proposed for wireless sensor networks in the literature (see, e.g., [31, 32]). We note, however, that PANEL does not need precise position information (see Section 6.3 for the related discussion); therefore, the inaccuracy of the positioning mechanism does not limit the applications of PANEL. Unlike the sensor nodes, the base stations may not necessarily be static, but they can be mobile and their presence can be sporadic.

We further assume that the sensor network consists of homogeneous sensors (in terms of resources). The sensor nodes are deployed in a bounded area, and this area is partitioned into geographical clusters. We aim at electing a single aggregator per cluster. The density of the network is large enough so that the nodes within each cluster are connected when they use maximum power for transmission. In other words, there exists a route between any pair of sensors of a given cluster that contains only sensors from that cluster. This assumption on the connectivity within a cluster is crucial to the correct operation of PANEL, and it can be satisfied by appropriately choosing the cluster size (given the deployment density of the network and the maximum power range of the nodes).

Finally, we assume that time is divided into epochs, and the nodes are synchronized such that each of them knows when a new epoch begins. If the nodes are equipped with GPS, then time synchronization is provided for free. Otherwise, additional mechanisms for time synchronization (see, e.g., [33, 34]) need to be implemented in the network in order to support PANEL.



+ Reference point
○ Sensor node
● Elected aggregator

Figure 1: Illustration of the geographical clustering in PANEL.

4. Operation of PANEL

In this section, we give a detailed description of the operation of PANEL. We start with a brief overview in order to introduce the components of PANEL, and then we present these components in detail in the subsequent subsections.

4.1. Overview. PANEL assumes that the sensor nodes are deployed in a bounded area, and this area is partitioned into geographical clusters. For simplicity, in this paper, we assume that the deployment area is a rectangle and that the clusters are equal-sized squares, as illustrated in Figure 1. We emphasize, however, that the ideas behind PANEL are general, and that PANEL could also be used for areas and cluster forms with more complex shapes.

The clustering is determined before the deployment of the network, and each sensor node is preloaded with the geographical information of the cluster which it belongs to. In our simplified case, each sensor node is preloaded with the coordinates of the lower-left corner of its cluster, as well as with the size d of the cluster. In addition, as we mentioned before, each node i is aware of its own geographical position $\vec{P}_i$.

At the beginning of each epoch, a reference point $\vec{R}_j$ is computed in each cluster j by every node in a completely distributed manner. In fact, the computation of the reference point depends only on the epoch number, and it can be executed by every node independently and locally. Once the reference point is computed, the nodes in the cluster elect the node that is the *closest to the reference point* as the aggregator for the given epoch (see Figure 1 for illustration).

The aggregator node election procedure needs communications within the cluster. PANEL takes advantage of these communications and uses them to establish routing tables for intra-cluster routing. In particular, at the end of the aggregator node election procedure, the nodes also learn the next hop towards the aggregator elected for the current epoch.

PANEL also includes a position-based routing protocol that is used in intercluster communications. As the nodes are aware of their geographical position, this seems to be a natural choice that does not result in additional overhead. The position-based routing protocol is used for routing messages from a distant base station or from a distant aggregator towards the reference point of a given cluster. Once the message enters the cluster, it is routed further towards the aggregator using the intra-cluster routing protocol based on the routing tables established during the aggregator node election procedure. Any position-based routing protocol can be integrated with PANEL; currently, we are using the Greedy Perimeter Stateless Routing (GPSR) protocol [35].

PANEL can also support reliable persistent data storage applications such as TinyPEDS [2]. Reliability can be achieved by replicating the data aggregated by the aggregator nodes at other aggregator nodes. For this purpose, the aggregator nodes need to be able to communicate with each other. The routing protocols of PANEL can support this by routing the messages containing the replicated data using PANEL's position-based intercluster routing protocol towards the reference point of the selected backup cluster, and then switching to the intra-cluster routing protocol of PANEL to deliver the data to the aggregator of that cluster.

Finally, we want to point out that in PANEL, the reference points of the clusters are recomputed and the aggregator election procedure is reexecuted in each epoch. This ensures load balancing in the sense that each node of the cluster can become aggregator with nearly equal probability. In addition, the nodes can accumulate information that they receive in the different epochs and use that for routing and intrusion detection purposes (see Section 6.1 for more details).

4.2. Reference Point Computation. In PANEL, the aggregator election begins with the computation of a reference point $\vec{R}_j$ in each cluster j. The input of this computation is the current epoch number e, which is assumed to be known by every sensor. The computation itself consists in calling a pseudorandom function H that maps e to a relative position $\vec{Q}$ inside the cluster. Formally, $H(e) = \vec{Q}$, where $\vec{Q} \in (-\delta d, d + \delta d) \times (-\delta d, d + \delta d)$, d is the size of the cluster, and $\delta < 0.5$ is a parameter which we will explain below. The reference point of cluster j is determined as $\vec{R}_j = \vec{O}_j + \vec{Q}$, where $\vec{O}_j$ is the position of the lower-left corner of cluster j.

The pseudorandom function H can easily be implemented with a cryptographic hash function. Moreover, the pseudorandomness of H means that the outputs produced by H for the consecutive epoch numbers look as a sequence of random positions. This ensures the load balancing property of PANEL.

Note that the above computation can be executed by every sensor independently and locally. In addition, the reference points of every past (and future) epoch can also be computed easily by anybody. This property is useful in applications where the sensor network provides persistent storage services by requiring the aggregator nodes of the different epochs to store the aggregated values that they

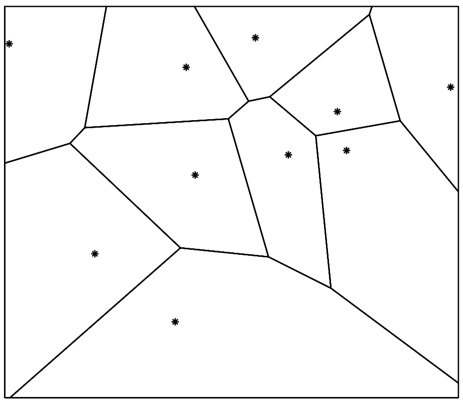


Figure 2: The Voronoi cells of the nodes in a cluster.

compute. In these applications, when looking for some data of a past epoch in a given cluster, one needs to send a query to the aggregator of that epoch. This requires the recomputation of the reference point of the given cluster in the given epoch.

Let us now explain why parameter δ is needed in the reference point computation, and how its value can be determined. Recall that in PANEL, the node that is the closest to the reference point of a given cluster is elected as aggregator for that cluster for the given epoch. Assuming that the nodes are deployed uniformly at random, and that the position of the reference point in each epoch is also selected uniformly at random, the probability that a given node becomes aggregator is determined by the size of the Voronoi cell of the node, and the size of the area within which the reference point is selected. For load balancing purposes, we would like that each node becomes aggregator with nearly the same probability, thus, we would like that the Voronoi cells of the nodes have approximately the same size.

Let us consider Figure 2 for illustration of the Voronoi cells of the nodes in a cluster. We can observe a "border effect" in this figure, namely, the size of the Voronoi cells of the nodes close to the edge of the cluster is larger than that of the nodes in the middle. One way to cancel this border effect out is to deploy the sensor nodes not at random, but following a given structure. This structure has to ensure that the sizes of the Voronoi cells of the different nodes are equal. For example, the ideal deployment position for node $i (i \leq n)$ in one dimension in $[0, t]$ is at $((2i - 1) \cdot t)/2n$, where n is the number of nodes. However, deploying the nodes by hand is often impossible (e.g., in many military applications); therefore, we set aside this solution. Even if we assume that the node deployment is fixed, we can effectively mitigate this border effect by adjusting the size of the area within which the reference point is selected, as with this, we can adjust the size of the Voronoi cells of the nodes on the edge of the cluster. Parameter δ expresses the magnitude of this adjusting operation in percent of the original cluster size d. For example, $\delta = -0.1$ means that on each side of the cluster the bounds are contracted by 10%.

It is not easy to determine an appropriate value for δ analytically due to the complexity of the computation of the

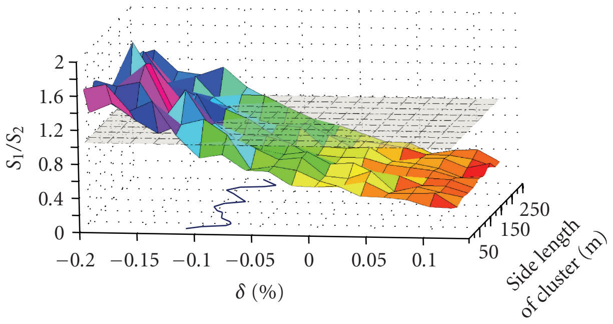


Figure 3: Determining the value of parameter δ by simulations.

size of the Voronoi cells. Therefore, we propose to determine its value by simulations. In Figure 3, on the z axis, we have the ratio between the average size S_1 of the bounded Voronoi cells (i.e., the cells close to the center of the cluster) and the average size S_2 of the unbounded Voronoi cells (i.e., the cells on the edge of the cluster) as a function of parameter δ and the cluster size given by the side length. The plane at $z = 1$ corresponds to the optimum, where the average sizes of the cells of the two types are equal. The intersection of this plane and the surface obtained by simulations is projected to the $z = 0$ plane. This projected curve gives the optimal value of parameter δ for different cluster sizes assuming that there are 10 nodes in the cluster. As one can see, the optimal value is usually between −0.12 and −0.07.

4.3. Aggregator Election Procedure. Once the reference points are computed, the nodes start the aggregator node election procedure. Each node i sets a timer, the expiration time of which is proportional to the distance $D(\vec{P}_i, \vec{R}_j)$ between the node's position $\vec{P}_i$ and the reference point $\vec{R}_j$ of its cluster. When this timer expires, the node broadcasts a message with maximum power in which it announces itself as the aggregator unless the node heard such an announcement from another node before its timer expired. The announcement message has the following format:

$$[\text{type} \mid \text{epoch} \mid \text{id} \mid \text{pos}] \tag{1}$$

where "type" is announcement, "epoch" is the current epoch number, and "id" and "pos" are the identifier and the position of the originator of the announcement, respectively.

When a node hears an announcement, it verifies whether the originator of the announcement is closer to the reference point than the node known to be the closest so far (which can be the node itself if it has not heard any announcements yet). If so, then the node records the originator of the announcement as the candidate aggregator, and rebroadcasts the announcement. Moreover, if the node still has its timer active, then it cancels it. Otherwise, the node silently discards the announcement. Announcements that belong to other clusters are also discarded in order to limit the propagation of an announcement within the cluster that it is concerned with.

```
Input:
    identifier $\mathrm{id}_{\mathrm{self}}$ and position $\vec{P}_{\mathrm{self}}$ of the node executing the algorithm
    parameters $\vec{O}_{\mathrm{self}}$ and $d$ of the cluster of the node executing the algorithm
    current reference point $\vec{R}_{\mathrm{self}}$ of the cluster and epoch number $e_{\mathrm{now}}$
    running time $T$ of the algorithm
Output:
    identifier $\mathrm{id}_{\mathrm{aggr}}$ and position $\vec{P}_{\mathrm{aggr}}$ of the elected aggregator node
set $\mathrm{id}_{\mathrm{aggr}} = \mathrm{id}_{\mathrm{self}}$;
set $\vec{P}_{\mathrm{aggr}} = \vec{P}_{\mathrm{self}}$;
set timer $t_0 = T$;
set timer $t_1 = f(D(\vec{P}_{\mathrm{self}}, \vec{R}_{\mathrm{self}}))$;
while timer $t_0$ is still active do
    wait until timer $t_1$ fires or an announcement $m$ is received;
    case timer $t_1$ fired:
        broadcast [announcement | $e_{\mathrm{now}}$ | $\mathrm{id}_{\mathrm{self}}$ | $\vec{P}_{\mathrm{self}}$] with max power;
    case an announcement $m =$ [announcement | $e$ | id | $\vec{P}$] is received:
        if the pair $(e, \mathrm{id})$ has been seen before then drop $m$;
        else if $e \neq e_{\mathrm{now}}$ or $\vec{P} \notin$ square $(\vec{O}_{\mathrm{self}}, d)$ then drop $m$;
        else if $D(\vec{P}, \vec{R}_{\mathrm{self}}) > D(\vec{P}_{\mathrm{aggr}}, \vec{R}_{\mathrm{self}})$ then drop $m$;
        else
            set $\mathrm{id}_{\mathrm{aggr}} = \mathrm{id}$;
            set $\vec{P}_{\mathrm{aggr}} = \vec{P}$;
            if timer $t_1$ is still active then cancel timer $t_1$;
            rebroadcast $m$ with max power;
end while
output $\mathrm{id}_{\mathrm{aggr}}$, $\vec{P}_{\mathrm{aggr}}$
```

Algorithm 1: The pseudocode of the aggregator election procedure of PANEL.

As the node that is the closest to the reference point sends its announcement first, there is a high chance that this will be the single announcement that is flooded inside the cluster. This means that in most cases, each node rebroadcasts a single message during the aggregator election procedure. In some cases, however, depending on the topology of the network, it may happen that more than one nodes send their announcements. In those cases, only the announcement originated by the node that is the closest to the reference point will “survive”, meaning that only that announcement will be received and recorded by every node in the cluster.

After some predefined time T, the aggregator node election phase is closed, and each node considers the recorded candidate aggregator as the aggregator for the current epoch. The value of T depends on the time needed for a flooded message to cover the largest possible distance within the cluster. This ensures that at the end of the aggregator election phase, each node must have received the announcement of the future aggregator.

The pseudocode of the aggregator election algorithm is given in Algorithm 1.

4.4. Routing. Strictly speaking, routing is not an integral part of aggregator node election protocols. Nevertheless, in PANEL, we make recommendations for the routing protocols that fit best PANEL’s design assumptions and operating principles. In particular, in PANEL, we envision two kinds of routing components: an intra-cluster routing protocol and an intercluster routing protocol.

The intra-cluster routing protocol is used to route a message to the aggregator of a given cluster if that message is already inside the cluster. This concerns, on one hand, the messages that contain the measurements of the sensors in the cluster. On the other hand, the intra-cluster routing protocol is also used to route messages from a distant source to the current aggregator or to any of the past aggregators of the cluster once those messages have reached the cluster. These messages include queries originating from a distant base station and backup messages originating from aggregators of distant clusters.

The intra-cluster routing protocol has a stronger connection to the operational details of PANEL, therefore, we specify it explicitly. However, we note that this specification serves as an example only; other kinds of routing protocols can also be integrated with PANEL.

The intra-cluster routing protocol of PANEL can take advantage of the fact that the nodes within the cluster communicate during the aggregator election procedure. In particular, announcement messages containing the identifier and the position information of their sources are flooded in the cluster. This can be used to set up backward pointers towards the sources of the announcement messages in the routing tables of the nodes. More specifically, in PANEL, every node that hears an announcement records the identifier and the position of the originator of the announcement as destination, it records the identifier of the node from which it received the first copy of the announcement as the next hop towards the recorded destination, and it computes and records the power level needed to transmit to this next hop node. The identifier of the next hop is obtained from the lower-layer (e.g., MAC) header of the message encapsulating the announcement. The computation of the required power level relies on the fact that the nodes transmit announcement messages with maximum power, and the receiving nodes can measure the power level with which they receive those messages.

Given the above information in the routing tables, and the identifier of the destination in the messages that are routed with the intra-cluster routing protocol, routing is quite straightforward. Each node in the cluster that receives such a message forwards it to the next hop associated to the message's destination in the routing table of the forwarding node with the corresponding power level. If no matching entry is found in the routing table, then the message is dropped. The positions of the destinations stored in the routing table are not used by the intra-cluster routing protocol; they are recorded to support the interoperation of the intercluster and the intra-cluster routing protocol, as we will explain later.

An important observation is that the aggregator election procedure described in Section 4.3 ensures that the announcement message of the future aggregator node of the current epoch is flooded in the *entire* cluster, and thus, *every* node in the cluster creates a routing entry (or updates an existing one) with the future aggregator as the destination. This means that later in the current epoch, every node in the cluster can forward messages towards this aggregator. Moreover, routing table entries are kept beyond the duration of the epoch in order to support the routing of queries that are destined to aggregators of past epochs.

The intercluster routing protocol is used to route messages to and from a distant cluster. These messages can be queries from and responses to a distant base station, as well as backup messages destined to distant aggregators that contain replicated data. We recommend to use a position-based routing protocol as the intercluster routing protocol for the following two reasons. First, PANEL already makes the assumption that the nodes are aware of their positions, and therefore, this position information can naturally be reused for routing purposes. Second, intercluster routing is concerned with messages that need to be routed (i) to the aggregator of a distant cluster or (ii) to a distant base station. Regarding case (i), in PANEL, the identifier of the aggregator node is not known explicitly outside the cluster, but, instead, one knows only the reference point to which the aggregator happens to be the closest node. Regarding case (ii), the query messages can contain the geographical position of the base station to which the responses should be sent back. Thus, in all cases, messages need to be routed towards a geographical position, and hence, position-based routing seems to fit best for intercluster routing in PANEL. Apart from being a position-based routing protocol, we do not restrict the choice for intercluster routing in PANEL.

The intercluster routing protocol is used together with the intra-cluster routing protocol in the following way. First of all note that messages from distant sources are always destined either to the current aggregator of a cluster or to one of the aggregators in the past. In particular, backup messages containing replicated data of another cluster are destined to the current aggregator of the backup cluster, whereas queries from the base station are usually destined to an aggregator in the past. Note also that, as we mentioned above, every node has routing table entries for the current and the past aggregators of its cluster. The interworking of the intercluster and intra-cluster routing protocols is based on these important observations.

Messages from distant sources do not contain the identifier of the targeted aggregator, but instead they contain the reference point to which the targeted aggregator is the closest node. When such a message reaches the target cluster, the first node that receives it looks into its routing table, and determines the identifier of the targeted aggregator by searching for the entry whose destination position is the closest to the reference point specified in the message. Once the identifier of the destination is determined, the intra-cluster routing protocol can be used to deliver the message. Once again, the correctness of this approach is based on the fact that *every* node in the cluster has an entry in its routing table for the current and *all* past aggregators of the cluster, and messages from distant sources are always destined to one of these aggregators.

5. Simulation Results

In this section, we study the aggregator node election and cluster formation capabilities of PANEL by means of simulations. Moreover, we compare the energy consumption of a network using PANEL as the aggregator node election protocol to the energy consumption of the same network when the aggregator election protocol is HEED [3]. We have chosen HEED as the algorithm to compare PANEL to as HEED is a probabilistic approach as well; moreover, it aims at energy efficiency too and it considers intra-cluster routing already, not just aggregator node election. HEED outperforms LEACH [8, 9] in terms of network lifetime and in ratio of energy dissipated for clustering, therefore, it is a better benchmark than LEACH. Finally, HEED is a widely known and well-understood aggregator election approach that is frequently referenced in the corresponding literature.

For the simulations, we have implemented both PANEL and HEED in NesC for TinyOS 2 [36] (i.e., the implementations are ready to run on real sensor nodes), and the simulations are made with the help of TOSSIM [37], the de-facto TinyOS simulator. The simulations focus on the aggregator node election phase of PANEL, as it is the main part of the protocol. As for the energy measurements, we based our solution on PowerTOSSIM 2. PowerTOSSIM 2 is the power measurement extension for TinyOS 2; however, it is still experimental and can be downloaded only from the official TinyOS 2 contrib repository [38]. As HEED needs energy information in runtime for the aggregator node election, we extracted the corresponding equations from the code of PowerTOSSIM 2 and integrated them into the code of HEED (and PANEL as well, to maintain equivalence in testing setup). In this way, each sensor node measures its own energy consumption. Note that this small computation consumes very limited energy and hence it does not influence the results significantly. We note that PowerTOSSIM 2 supports only mica2 nodes [39] which use the Chipcon CC1000 [40] radio transceiver and the ATmega128L [41] microcontroller.

The simulation settings are the following. We assume that time is divided into epochs, and each epoch consists in an aggregator node election phase and five measuring/data transmission phases, called rounds. In the aggregator node election phase, the nodes select the cluster head in each cluster, while in the rounds, the cluster member nodes send measurement information to the cluster head. Both the aggregator node election phase and the rounds have at most 10 seconds to finish. We ran the simulations for 100 epochs (i.e., for 6000 seconds). The tested topologies are the same for PANEL and HEED. We considered 40 uniformly randomly placed nodes in a square-shaped field consisting in 4 equal shaped clusters, where the node density is controlled through the size of the field: the highest density is realized in the smallest field of $50 \times 50\,\mathrm{m}^2$, while the smallest density is realized in the largest field of $500 \times 500\,\mathrm{m}^2$. We simulated 20 different topologies for each of the different densities. In case of HEED, the value of parameter $p_{\min}$ is set to $5 \cdot 10^{-4}$, and the cost model is AMRP (see [3]) with the modification that it does not depend on the total number of nodes in the cluster, but only on the number of nodes that are in reachable distance with radio communication. In case of PANEL, δ is set to 0, as this choice corresponds to the worst case (i.e., no border effect mitigation).

The first question we analyze is the consistence of aggregator node election in case of PANEL and HEED. We call the aggregator election consistent if only one cluster head (i.e., aggregator node) is elected in each cluster. This consistence is the optimal case; however, due to message losses, interferences and possible lack of connectivity, this ideal case rarely happens (see the related discussion in Section 6.2). Nevertheless, the closer the number of cluster heads is to the number of clusters, the more consistent is the aggregator election. In Figure 4, one can see the average number of cluster heads as a function of the field size (i.e., node density). The horizontal axis corresponds to the side length of the field (as the field is assumed to be square shaped, both sides of the field are of the same length), while the vertical axis corresponds to the average number of cluster heads. The solid curve corresponds to PANEL, while the dashed curve corresponds to HEED. The measurements are made only on the indicated field sizes (see the ticks on the horizontal axis), the lines connecting the points are only to emphasize the characteristics of the changes when heading to lower node densities. The presented measurements are averages of 20 different topologies for each field size, and

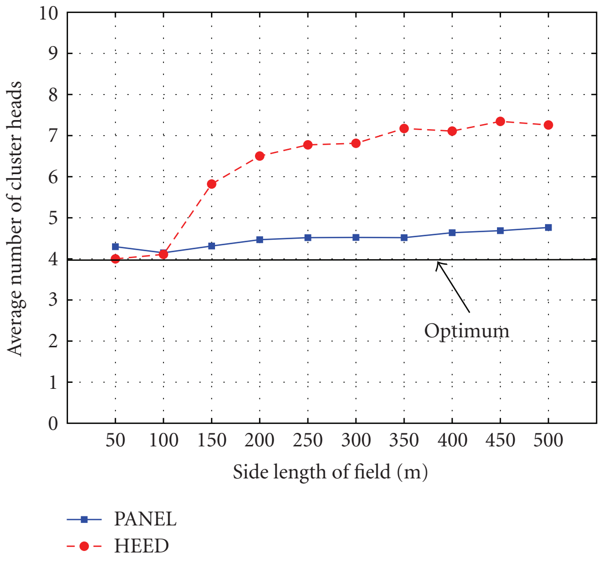


Figure 4: Average number of cluster heads as a function of the field size.

100 epochs are simulated on each topology. (This way of visualization will be the same in the upcoming figures as well.)

As one can see, PANEL ensures a more consistent aggregator node election than HEED. The line corresponding to the optimum is at $y = 4$ (as we have 4 clusters), and PANEL approaches it quite well, while in case of HEED, the average number of cluster heads goes over 7 for lower node densities. We note that for very high node densities (i.e., field sizes of $50 \times 50\,\mathrm{m}^2$ and $100 \times 100\,\mathrm{m}^2$) HEED performs slightly better than PANEL, but for average and low node densities, the results of PANEL are much better than the results of HEED. The reason for this outcome lies in the very nature of the algorithms: PANEL can better propagate the cluster head advertisements since these messages always get rebroadcast as a node receives and accepts them. However, in case of HEED, the advertisements are not rebroadcast (i.e., only one broadcast is sent by the advertiser node that all the other nodes have to receive), so there is a higher chance for a node to miss such a message and declare itself as a cluster head, thus raising the average number of cluster heads.

The time spent for electing the cluster head is another important question. The less time an algorithm needs for cluster head election, the more time it has for other purposes. We define the cluster head election time as the length of the time interval beginning with the new epoch start, and ending when all the nodes have a finalized view about which node is the cluster head (and which node is the next hop towards this cluster head). In Figure 5, one can see the average cluster head election time (vertical axis) as a function of the field size (horizontal axis). The solid curve corresponds to PANEL, while the dashed curve corresponds to HEED.

The results of Figure 5 are not surprising. Both PANEL and HEED have 10 seconds according to the simulation setup for cluster head election in each epoch, and in HEED, this time is consumed as the finalization of the links is only

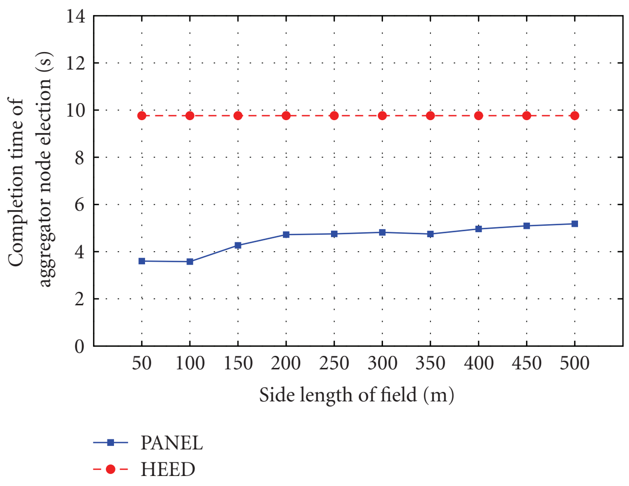


Figure 5: Average cluster head election time as a function of the field size.

performed after some number of advertisement iterations is already performed. The required number of such iterations in HEED is controlled by $p_{\min}$, however, we have chosen $p_{\min}$ for five times higher than in the examples in [3] in order to lower the number of required iterations (i.e., from 15 to 12). Further lowering the number of iterations could possibly distort the cluster head election process in HEED as its correct operation—especially its energy balancing property—relies on the fact that higher remaining energy nodes have less (approx. 6) iterations to perform than lower remaining energy nodes, and thus, the latter nodes are able join to the clusters established by the former nodes. However, in PANEL, the 10 seconds time window for cluster head election is not fully consumed: PANEL needs less than 6 seconds for electing the cluster head (and for establishing the multi-hop route for each node towards this cluster head). The reason for this is that, in the optimal case, all the nodes have to send one broadcast message and extract the required information from the received messages, thus, the cluster head election can end without further iterations.

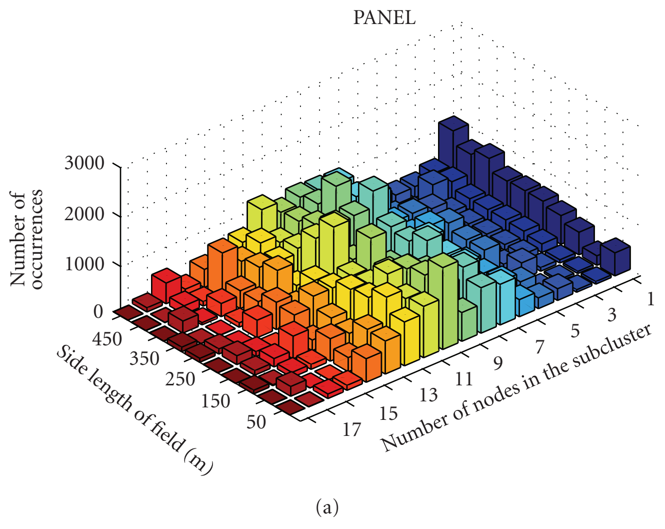
Having seen that the number of cluster heads is usually not optimal on average (see Figure 4), it is very important to detail the impact of this result on the number of nodes in the partitioned clusters (i.e., subclusters). As we have 40 nodes deployed on the field and 4 clusters, the optimal number of nodes in a cluster is 10. However, as the nodes are randomly deployed, it often happens that there are fewer than 10 nodes in some clusters (and more than 10 in the others), but this does not influence our average expectation. In Figure 6, one can see the histogram of the number of nodes in the subclusters as a function of the field size. The x-axes corresponds to the field size, the y-axes corresponds to the number of nodes in the subcluster, while the z-axes corresponds to the number of occurrences.

Figure 6 clearly shows that while HEED produces a significant amount of clusters with few nodes (i.e., the rearmost bars are high), PANEL usually produces clusters of near optimal number of nodes (i.e., the bars near to the $y = 10$ axis are relatively high). This again emphasizes a good feature of PANEL: even if the number of cluster heads gets above the optimal value, the number of nodes in a subcluster remains high. On the opposite, HEED produces a high amount of clusters with very few nodes, thus subdivides the network into small parts that are hard to maintain. For the sake of better comparison, we show three sections of Figure 6 in Figure 7.

In Figure 7, the horizontal axes correspond to the number of nodes in the subclusters, while the vertical axes correspond to the number of occurrences. The solid curve corresponds to PANEL, while the dashed curve corresponds to HEED, and the three subfigures correspond to different field sizes, namely $50 \times 50\,\text{m}^2$, $250 \times 250\,\text{m}^2$ and $500 \times 500\,\text{m}^2$, respectively. The tendency of the behaviour of the two algorithms is clear: while HEED behaves very well in terms of distribution of the number of nodes in the subclusters for high node densities, it produces many clusters with few nodes as the node density goes lower. On the contrary, PANEL behaves quite well in the same comparison even for small node densities. The reason for this difference stems from the fact that in HEED, the cost model can retain nodes from joining to a common cluster head when the node density is small, as further lying cluster heads have higher costs than closer-lying ones. Another component of the problem is the high average number of cluster heads (see Figure 4): HEED elects more cluster heads on average, which also leads to network partitioning. As the connectivity becomes weaker (i.e., for lower node densities, larger fields), this behaviour becomes more emphasized, however, in PANEL, the number of clusters with few nodes stays low even for low node densities.

All the same, cluster head election makes no sense without an application that employs the newly elected cluster heads. This justifies our simulation scenarios that comprise not just cluster head election, but data message sending as well. Instead of measuring just the energy spent for cluster head election, in Figure 8(a), we show the total average energy consumed by PANEL and HEED. Here again, the horizontal axis corresponds to the field size, while the vertical axis corresponds to the average energy consumption. The solid curve corresponds to PANEL, while the dashed curve corresponds to HEED. The whiskers show the 95% confidence interval of the corresponding values.

As one can see, PANEL consumes less energy in total than HEED, independently from the node density. Moreover, the whiskers in Figure 8(a) show that the energy consumption of PANEL can be forecasted more precisely than the energy consumption of HEED, as the 95% confidence intervals of the energy consumption of PANEL are more narrow than that of HEED. This property is confirmed by Figure 8(b), which shows that the standard deviation of the average energy consumption is much higher in case of HEED than in case of PANEL. (The whiskers in Figure 8(b) correspond to the 95% confidence interval of the standard deviation of the energy consumption.) We emphasize that the number of rounds (i.e., the amount of data message sending) highly influences our results: the rounds are where PANEL is more energy efficient than HEED, thus, increasing the number of



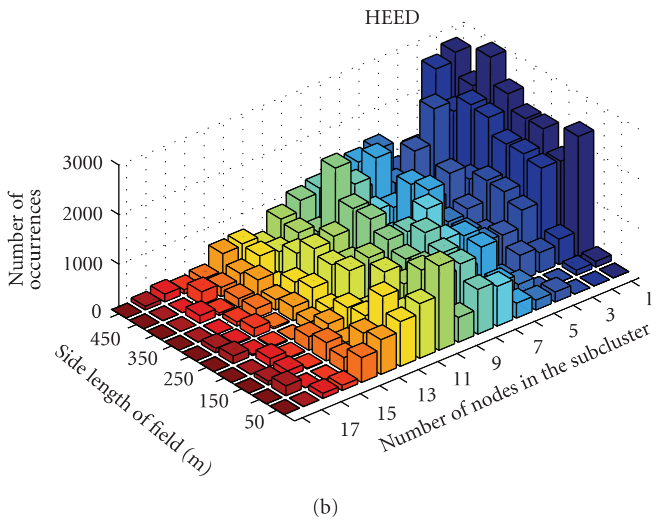


Figure 6: Histogram of the number of nodes in the subclusters.

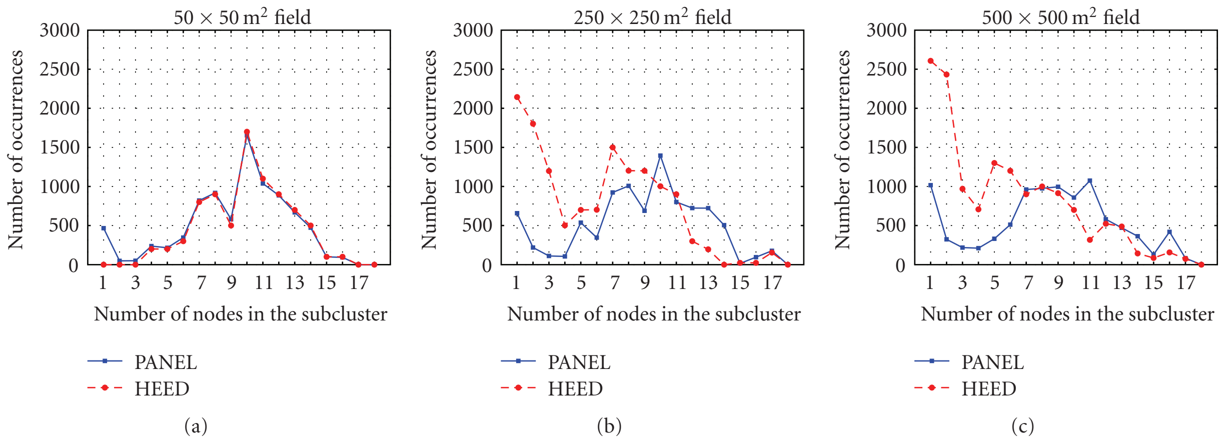


Figure 7: Histogram of the number of nodes in the subclusters for field sizes of $50 \times 50\,\text{m}^2$, $250 \times 250\,\text{m}^2$ and $500 \times 500\,\text{m}^2$.

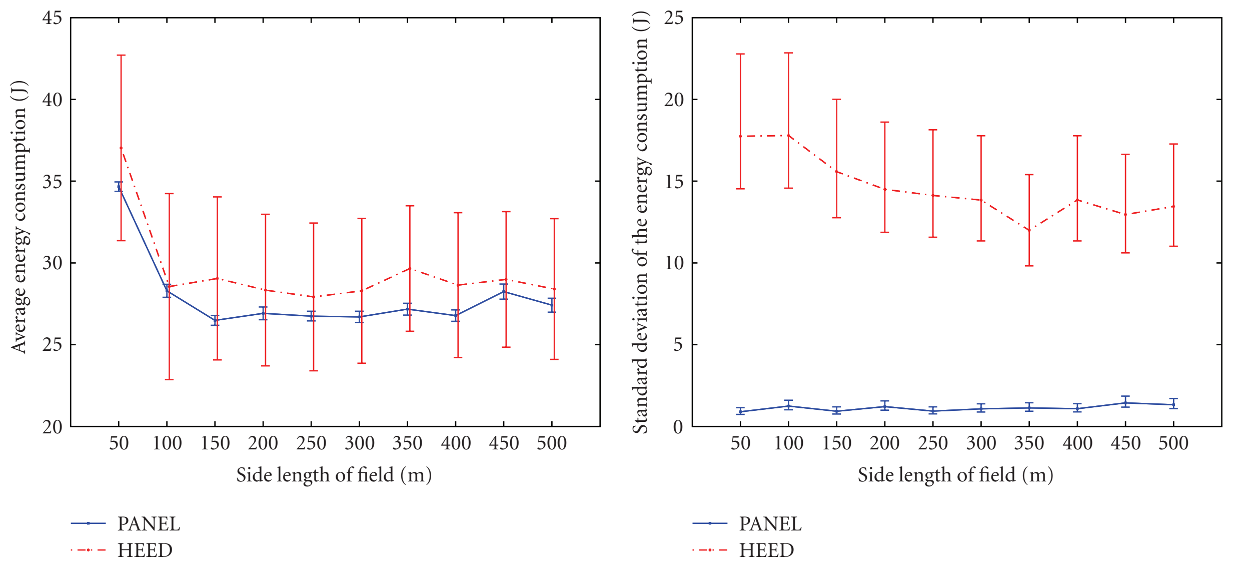
rounds beyond 5 would result in even better performance of PANEL with respect to HEED. The number of rounds depends on the actual application, however, we believe that having 10, 20, 50 or more rounds can be the general case. In these energy simulations we tried to show how PANEL and HEED works in worst case situations, and we conclude that PANEL is more energy efficient that HEED in our scenarios, independently from the node density.

6. Extensions of PANEL

We complete the description of PANEL in this section by discussing three possible extensions related to its operation: the security of PANEL, the problem of disconnected clusters, and the required accuracy of the node's position information.

6.1. Security. There are several ways how an adversary can spoil the operation of PANEL. In the following, we detail these attacks and propose countermeasures against them. We assume that the attacker is able to capture and reverse-engineer one or more nodes, thus, the attacker is able to control these nodes and has knowledge about all the information stored in these nodes (e.g., secret keys, measurement data, etc.).

The most straightforward type of attack aims at distorting the aggregate at the cluster head. To achieve this, an attacker can either (i) modify the environment of the attacked sensor node, or (ii) capture the sensor node and alter the measured values as desired. Altering the measured parameter of the environment in the first attack means, for example, lighting a match near to a temperature sensor or flashing with a flashlight near to a photometer sensor; these outsider attacks can totally falsify the measurement result of the sensor node. Capturing a sensor node in the second case requires more knowledge from the adversary, but he can gain full control over the sensor node. Both of these attacks can be circumvented using a statistical sample filtering approach at the cluster head (like, e.g., [42]). (We note that cryptography cannot help here as these attacks cannot be detected with cryptographic tools.)



(a) Total average energy consumption as a function of the field size

(b) Standard deviation of the total average energy consumption as a function of the field size

Figure 8: Total energy consumption comparison of PANEL and HEED.

In order to achieve his goal (i.e., to distort the aggregate), the adversary can also (iii) force the captured node to alter the data field of a forwarded message that comes from another node, or (iv) send false measurement data in the name of other nodes to the cluster head. In case of (iii), the captured node must be on the routing path that one or more of the nodes use to send their measurements to the cluster head. As these messages are not protected, a captured node that is a forwarder can modify the content of these messages. In case of (iv), the captured node tries to confuse the cluster head by sending messages in the name of other nodes; when the cluster head later receives the valid messages from the well-behaving nodes it cannot determine which of the received messages are valid. While both of the latter attacks can be handled more or less with the previously mentioned technique, a more appropriate tool against these attacks is cryptographic integrity protection (in case of (iii)), or authentication (in case of (iv)). For these, the nodes need, for example, a public-private key pair and they have to sign their messages using the private key, moreover, they have to attach their public key to the message after it was signed. (We note that assuming public-key cryptography in sensor networks is not far fetched according to [43].) This, on one hand, prevents a captured node to alter the measurement data in the messages as the captured node cannot regenerate the signature on the message after the modification. On the other hand, a captured node cannot send messages in the name of other nodes as, again, the captured node cannot generate a valid signature, because it does not know the private key of other nodes. However, it could be possible that an attacker invents public-private key pairs and generates cryptographically sound messages. We anticipate that this attack is handled by using the extensions proposed later in this section (i.e., identifier signing by the base station).

Signing and thus authenticating the messages also helps in the case when the attacker aims at interfering with the cluster head election process. In this attack, the attacker tries to send an advertisement message in the name of a node that would not become aggregator based on the position of the reference point. The nodes that hear this fake advertisement message would rebroadcast it until a node detects that its own position is closer to the reference point than the position in the advertisement. After this, the latter node would rebroadcast its own advertisement and the whole cluster would know which node is the real cluster head. However, the numerous message broadcasts consumes a high amount of energy, and thus, this attack should not be underestimated. Signing not just the data messages but the advertisement messages as well protects against this attack as, in this case, an attacker cannot send a valid advertisement message in the name of other nodes as it does not know the other nodes' private key. A node that receives an advertisement message that is not correctly signed can easily drop it and further wait for a valid advertisement.

A typical attack against aggregator node election protocols is to manipulate the execution in such a way that the nodes controlled by the adversary become aggregators more frequently than they should. In this way, the adversary can collect information from the network easier, as nodes send their sensor readings to the aggregators. In PANEL, such an attack can be perpetrated using fake information in the announcement message in the aggregator node election phase by a captured node that uses (i) its correct identifier, but fake position information, or (ii) a fake identifier along with fake position information. Moreover, (iii) the adversary can deploy new nodes at desired positions. In the first case, the captured node only alters its position information and reports its correct identifier very close to the current reference point in each epoch. In the second case, the captured node invents a new identifier along with a position very close to the current reference point and reports this pair in the announcement. In the third case, however, the attacker does not have to capture a node, he only has to deploy a new one close to the reference point (the attacker can do this as he can calculate the position of the reference point in advance).

PANEL can be easily extended with security measures to prevent even these misdeeds. First of all, the base station can use public-key cryptography and sign the nodes' identifier with its private key, and load the corresponding public key on the sensors before deployment. Using this signed identifier, a node that receives an announcement message can check whether it contains a valid identifier or not using the public key of the base station, but no one except the base station is able to generate new identifiers. With this technique, one can detect the cases (ii) and (iii), therefore, the only remaining way to attack the aggregator node election phase is case (i). One can thwart attack (i) by allowing the nodes to keep in their routing tables the position information of the other nodes from which they have already heard an announcement. This information can be kept in the routing tables even beyond the duration of an epoch. Therefore, the nodes can detect if a captured or corrupted node tries to report itself at different positions in different epochs. If the above attack is detected, the detector node can flood an alarm message in the cluster including the cheating node's identifier and the proof for the cheating, that is, the two advertisement messages with the same identifier and different position information, both signed by the cheating node. In case of reception of such an alarm message the nodes can check the proof of the cheating and exclude the related identifier from further cooperation. We note that it is important that all the nodes exclude the cheating node in order to maintain a consistent view of the cluster. Without this requirement, some nodes could elect the cheating node later as aggregator, while some nodes could not, and this could result in cluster fractioning.

Another straightforward attack against the data collection part of PANEL is the selective message dropping attack. A captured forwarder node can selectively drop the messages that it receives instead of forwarding them. With this attack, however, the attacker can only slightly mislead the aggregate. For example, if the aggregator function is the average, the median, or the min/max, then the gain of the attacker is that the final aggregate is based on fewer measurements than in the unattacked case, which usually does not influence the result significantly (assuming that the number of dropped messages is not high). Moreover, each dropped measurement alters the aggregation result, but in case of the average and the median, the attacker does not know in advance whether a dropped measurement results in increasing or in decreasing the aggregate, and therefore, the strength of this kind of attack is limited as well. We believe that the very nature of sensor networks, namely that they are distributed and comprise of a high number of nodes, is enough to mitigate this type of attack.

In summary, to enhance the security of PANEL, we propose to use (i) node identifiers that are signed by the base station in order to prevent the acceptance of forged identifiers, (ii) digital signatures in order to authenticate the senders and protect the integrity of the messages, (iii) the technique of remembering the nodes' identifier and position in order to prevent the captured nodes from virtually changing their position, and (iv) a sample filtering techniques like those in [42] in order to mitigate attacks aiming at injecting false measurements.

6.2. Disconnected Clusters. A crucial assumption of PANEL is that the nodes within a cluster form a connected subnetwork. If this assumption is not satisfied, and the subnetwork within a cluster is partitioned, then some nodes will not hear the announcement of the node closest to the reference point, and they will elect another node as aggregator. More specifically, in this case, as many aggregators are elected in the cluster as many partitions the subnetwork has.

Connectivity within every cluster can be ensured by appropriately choosing the cluster size given the node density of the network. The smaller the clusters are, the more likely is that the subnetworks of the clusters will be connected given a particular node cardinality. We observe, however, that the node density may decrease during the lifetime of the network because some nodes may exhaust their batteries and die. One solution would be to introduce new nodes in the network in order to keep the node density constant. Another solution is to extend the area in which an announcement is flooded beyond the borders of the corresponding cluster. For instance, the announcement can also be flooded in the neighboring clusters. This would increase the probability that each node in the corresponding cluster receives the announcement even if the subnetwork within that cluster is partitioned, because those partitions may be connected through the neighboring clusters. The downside of this approach is the increased energy consumption of the nodes.

This latter approach is acceptable, if the nodes in the neighboring clusters are ready to route even the data messages of the disconnected clusters towards the affected cluster head. However, assuming that the queries from the base station are usually interested in aggregates of the past, the following solution can be more energy efficient. During the aggregator node election phase, we do extend the area in which an announcement is flooded, and we tolerate that multiple cluster heads are elected. These cluster heads collect the measurements of the sensors, and send the aggregated measurements to a backup cluster head as usual, also including their position information into the message. The backup cluster head can detect that multiple cluster heads were elected in a cluster when it receives multiple backups originated from the same cluster. This, on the one hand, can be reported to the base station by the backup cluster head, and, on the other hand, the subaggregates can be aggregated in one message by the backup cluster head again, and sent back to the originator cluster heads using solely position-based routing in order to recover the consistency inside the cluster.

This latter solution is efficient in terms of communication overhead, but sometimes it is not applicable. For example, in case of the average, it works fine, as the average of the averages of two subsamples is equal to the average of the sample consisting in the two subsamples. However, in case of the median, it does not work. Therefore, for such aggregation functions that need the whole sample to produce the correct output, we propose to use the following method. At first, we again require the aggregator nodes to include their position information in the backup messages. Then, as the backup cluster head detects the malfunctioning as already detailed, it chooses a "final aggregator" (i.e., a node among the cluster heads of the partitioned cluster that will finally perform the aggregation), and it informs the originator nodes about the disconnectivity and about the identity of the final aggregator. The packet sent to the originator nodes has to contain the identifier and position information of the final aggregator, while the packet sent to the final aggregator has to contain the identifier and position information of all the originators. All of these packages have to be sent using solely position-based routing. When the originator nodes receive such a packet, they send all of the collected measurements (i.e., not just the aggregates, but all of the measurements of all of the nodes in the subcluster in the epoch) to the final aggregator using position-based routing, which can then produce a proper aggregate based on the received sample. Finally, the final aggregator sends back the produced aggregate to the originator nodes using solely position-based routing. At the reception of this latter packet, the originators replace their previous aggregate with the received one. With this technique, even the partitioned subnetworks will have a consistent picture of their cluster, independently from the applied aggregator function. Moreover, queries will receive correct answers independently from the queried cluster head.

6.3. Virtual Coordinates. The operation of PANEL relies on the assumption that the nodes are aware of their geographic positions. This means that some positioning mechanism needs to be implemented in the network in order to support PANEL. Note, however, that the aggregator node election procedure itself does not require accurate position information; indeed, the same procedure would work along with the intra-cluster routing with virtual positions invented by the nodes themselves once and forever at the beginning of the operation of the network.

After all, interpreting the output of the pseudorandom function H as the position of a reference point is also an arbitrary choice; it could also be mapped to the identifier space, and the protocol could elect the node whose identifier is the closest to the "reference identifier" as the aggregator for the current epoch.

Besides the aggregator node election procedure, another component of PANEL, the intercluster routing protocol also uses the position information of the nodes. Therefore, the required accuracy of the position information is determined by the position-based intercluster routing protocol used in PANEL. Note, however, that one may consider replacing the position-based intercluster routing protocol with a nonposition-based protocol in order to further decrease the dependency of PANEL on the accuracy of the positioning mechanism.

7. Conclusion

We described PANEL: a position-based aggregator node election protocol for wireless sensor networks. The novelty of PANEL with respect to other aggregator node election protocols is that it supports asynchronous sensor network applications where the sensor readings are fetched by the base stations after some delay. In particular, the motivation for the design of PANEL was to support reliable and persistent data storage applications, such as TinyPEDS.

PANEL uses the position information of the nodes to determine which of them should become aggregator. PANEL ensures load balancing, meaning that each node has nearly the same chance to become aggregator, and it supports intra and intercluster routing allowing sensor-to-aggregator, aggregator-to-aggregator, base station-to-aggregator, and aggregator-to-base station communications.

Besides describing the operation of PANEL, we also evaluated its efficiency by means of simulations. In particular, we compared the cluster formation capabilities and the energy consumption of PANEL to those of HEED: an aggregator node election protocol well-known from the literature. Our results show that PANEL behaves better than HEED in both comparisons.

Acknowledgments

The work described in this paper is based on results of the IST FP6 STREP UbiSec&Sens (http://www.ist-ubisecsens.org/). UbiSec&Sens receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. The presented work has also been partially supported by the Hungarian Scientific Research Fund (contract number T046664). The first author has been partially supported by the Hungarian Academy of Sciences through the Bolyai Research Fellowship. The second author has been partially supported by Ericsson through the HSN Lab. Last, but not least, the authors are thankful for Gergely Ács and Attila Faddi for their help and support on this work. This work was done while the second author was with BME.

References

[1] L. Buttyán and P. Schaffer, "PANEL: position-based aggregator node election in wireless sensor networks," in *Proceedings of the 4th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS '07)*, pp. 1–9, Pisa, Italy, October 2007.

[2] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "TinyPEDS: tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 1073–1089, 2007.

[3] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, pp. 629–640, Hong Kong, March 2004.

[4] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," *IEEE Network*, vol. 20, no. 3, pp. 20–25, 2006.

[5] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[6] V. Mhatre and C. Rosenberg, "Homogeneous vs heterogeneous clustered sensor networks: a comparative study," in *Proceedings of IEEE International Conference on Communications (ICC '04)*, pp. 3646–3651, Paris, France, June 2004.

[7] V. Mhatre and C. Rosenberg, "Design guidelines for wireless sensor networks: communication, clustering and aggregation," *Ad Hoc Networks*, vol. 2, no. 1, pp. 45–63, 2004.

[8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, January 2000.

[9] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[10] S. Lindsey and C. S. Raghavendra, "PEGASIS: power-efficient gathering in sensor information systems," in *Proceedings of the IEEE Aerospace Conference*, vol. 3, pp. 1125–1130, 2002.

[11] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.

[12] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, pp. 1713–1723, San Francisco, Calif, USA, April 2003.

[13] M. Ye, C. F. Li, G. H. Chen, and J. Wu, "EECS: an energy efficient clustering scheme in wireless sensor networks," in *Proceedings of the 24th IEEE International Performance Computing and Communications Conference (IPCCC '05)*, pp. 535–540, Phoenix, Ariz, USA, April 2005.

[14] S. Soro and W. B. Heinzelman, "Prolonging the lifetime of wireless sensor networks via unequal clustering," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS '05)*, pp. 1–8, Denver, Colo, USA, April 2005.

[15] C. Li, M. Ye, G. Chen, and J. Wu, "An energy-efficient unequal clustering mechanism for wireless sensor networks," in *Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '05)*, pp. 597–604, Washington, DC, USA, November 2005.

[16] G. Gupta, "Load-balanced clustering of wireless sensor networks," in *Proceedings of IEEE International Conference on Communications (ICC '03)*, pp. 1848–1852, Anchorage, Alaska, USA, May 2003.

[17] G. Gupta and M. Younis, "Performance evaluation of load-balanced clustering in wireless sensor networks," in *Proceedings of the 10th International Conference on Telecommunications (ICT '03)*, 2003.

[18] M. Younis, M. Youssef, and K. Arisha, "Energy-aware management for cluster-based sensor networks," *Computer Networks*, vol. 43, no. 5, pp. 649–668, 2003.

[19] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '03)*, vol. 3, pp. 1525–3511, New Orleans, La, USA, March 2003.

[20] K. Kalpakis, K. Dasgupta, and P. Namjosh, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[21] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, "Optimal energy aware clustering in sensor networks," *Sensors*, vol. 2, no. 7, pp. 258–269, 2002.

[22] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor networks," in *Proceedings of the 3rd Annual Communication Networks and Services Research Conference*, pp. 255–260, Halifa, Canada, May 2005.

[23] P. Popovski, F. H. P. Fitzek, H. Yomo, T. K. Madsen, and R. Prasad, "MAC-layer approach for cluster-based aggregation in sensor networks," in *Proceedings of the International Workshop on Wireless Ad-Hoc Networks (IWWAN '04)*, pp. 89–93, Oulu, Finland, June 2004.

[24] M. Demirbas, A. Arora, and V. Mittal, "FLOC: a fast local clustering service for wireless sensor networks," in *Proceedings of Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS '04)*, 2004.

[25] F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing newly deployed ad hoc and sensor networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM '04)*, pp. 260–274, Philadelphia, Pa, USA, 2004.

[26] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical conrol in multi-hop wireless networks," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, Anchorage, Alaska, USA, April 2001.

[27] H. Chan and A. Perrig, "ACE: an emergent algorithm for highly uniform cluster formation," in *Proceedings of the 1st European Workshop on Sensor Networks*, pp. 154–171, Berlin, Germany, January 2004.

[28] M. Sirivianos, D. Westhoff, F. Armknecht, and J. Girao, "Non-manipulable aggregator node election protocols for wireless sensor networks," in *Proceedings of the 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '07)*, pp. 1–10, Limassol, Cyprus, April 2007.

[29] D. Liu, "Resilient cluster formation for sensor networks," in *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*, p. 40, Toronto, Canada, June 2007.

[30] J. N. Al-Karaki, R. Ul-Mustafa, and A. E. Kamal, "Data aggregation in wireless sensor networks—exact and approximate algorithms," in *Proceedings of the Workshop on High Perfomance Switching and Routing (HPSR '04)*, pp. 241–245, Phoenix, Ariz, USA, April 2004.

[31] M. Maróti, B. Kusy, G. Balogh, et al., "Radio interferometric geolocation," in *Proceedings of the ACM Workshop on Security*

of Ad Hoc and Sensor Networks (SENSYS '05), pp. 1–12, San Diego, Calif, USA, November 2005.

[32] S. Čapkun, M. Hamdi, and J.-P. Hubaux, "GPS-free positioning in mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 157–167, 2002.

[33] S. Ganeriwal, S. Capkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proceedings of the ACM Workshop on Wireless Security (WiSe '05)*, pp. 97–106, Cologne, Germany, September 2005.

[34] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 264–277, Alexandria, Va, USA, November 2006.

[35] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 243–254, New York, NY, USA, 2000.

[36] "TinyOS 2," http://www.tinyos.net/tinyos-2.x/doc/.

[37] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pp. 126–137, Los Angeles, Calif, USA, November 2003.

[38] TinyOS Alliance, "PowerTOSSIM for TinyOS 2.x," http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/cedt/

[39] XBow Corporation, "Mica2 Datasheet," https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf.

[40] Chipcon Corporation, "CC1000 Datasheet," http://www.chipcon.com/files/CC1000_Data_Sheet_2_2.pdf.

[41] ATMEL Corporation, "ATmega128L Datasheet," http://www.datasheetcatalog.com/datasheets_pdf/A/T/M/E/ATMEGA128L.shtml.

[42] L. Buttyán, P. Schaffer, and I. Vajda, "RANBAR: RANSAC-based resilient aggregation in sensor networks," in *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06)*, pp. 83–90, Alexandria, Va, USA, October 2006.

[43] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the 4th ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '06)*, pp. 169–176, Alexandria, Va, USA, October 2006.