PANEL: Position-based Aggregator Node Election in Wireless Sensor Networks*

Levente Buttyán Péter Schaffer
Laboratory of Cryptography and Systems Security (CrySyS)
Budapest University of Technology and Economics (BME)
{buttyan, schaffer}@crysys.hu

Abstract

In this paper, 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. PANEL ensures load balancing, and it supports intra- and inter-cluster routing allowing sensor to aggregator, aggregator to aggregator, base station to aggregator, and aggregator to base station communications. We also present simulation results showing that PANEL is very energy efficient.

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 real-time using multi-hop 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 real-time operation is indispensable. Examples of asynchronous applications include habitat monitoring systems and agricultural applications such as wine yard monitoring where real-time 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 for the dynamic re-assignment of the aggregator role.

In this paper, 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 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

*The work described in this paper is based on results of the IST FP6 STREP UbiSec&Sens (www.ist-ubisecsens.org). The presented work has also been partially supported by the Hungarian Scientific Research Fund (contract number T046664) and the HSN Lab.

1-4244-1455-5/07/$25.00

supports asynchronous applications.

In particular, the motivation for the design of PANEL was to support TinyPEDS (Tiny Persistent Encrypted Data Storage) [5], 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 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 introduce the general assumptions that we based the design of PANEL upon. In Section 3, we describe the operation of PANEL and discuss some of its features. In Section 4, we present our simulation-based analysis of PANEL, and in particular, a comparison of its performance with that of LEACH [6], an aggregator node election protocol well-known from the literature. Finally, in Section 5, we report on some related work, and in Section 6, we conclude the paper.

2. 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., [9, 12]). We note, however, that PANEL does not need precise position information (see Subsection 3.5 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 need to be implemented in the network in order to support PANEL.

3. 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.

3.1. 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 the clusters are equal sized squares, as illustrated in Figure 1. We emphasize, however, that the ideas behind PANEL are general, and 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 pre-loaded with the geographical information of the cluster which it belongs to. In our simplified case, each sensor node is pre-loaded 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).

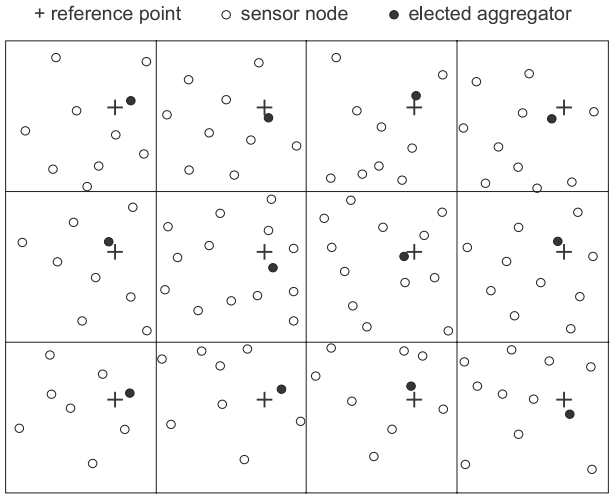

Figure 1. Illustration of the geographical clustering in PANEL

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 inter-cluster 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 experimenting with the Greedy Perimeter Stateless Routing (GPSR) protocol [7].

Finally, we want to point out that in PANEL, the reference points of the clusters are re-computed and the aggregator election procedure is re-executed 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 Subsection 3.5 for more details).

3.2. 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 pseudo-random 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 $\delta < 1$ 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 pseudo-random function H can easily be implemented with a cryptographic hash function. Moreover, the pseudo-randomness 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 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 re-computation 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" on this figure, namely, the size of the Voronoi cells of the nodes close to the edge of the

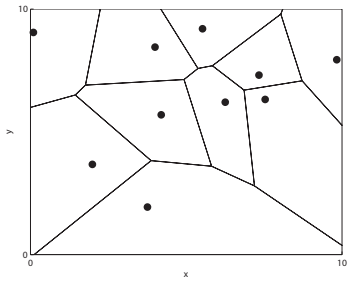

Figure 2. The Voronoi cells of the nodes in a cluster

cluster is larger than that of the nodes in the middle. We want to cancel this border effect out by somehow adjusting the size of the Voronoi cells and that of the area within which the reference point is selected. As a matter of fact, the Voronoi cell of a node surrounded by other nodes is fixed, but we can adjust the size of the Voronoi cells of the nodes on the edge of the cluster by re-sizing the area within which the reference point is selected. Parameter δ expresses the magnitude of this re-sizing operation in percent of the original cluster size d. For example, $\delta = -0.03$ means that on each side of the cluster the bounds are contracted by 3%.

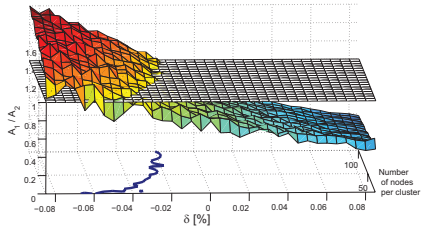

Figure 3. Determining the value of parameter δ by simulations.

It is not easy to determine an appropriate value for δ analytically due to the complexity in the computation of the size of the Voronoi cells. Therefore, we proposed to determine its value by simulations. In Figure 3, on the z axis, we have the ratio between the average size of the bounded Voronoi cells (i.e., the cells close to the center of the cluster) and the average size of the unbounded Voronoi cells (i.e., the cells on the edge of the cluster) as a function of parameter δ and the number of nodes per cluster. 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 number of nodes in the cluster. As one can see, the optimal value is usually between -0.04 and -0.02.

3.3. 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:

$$[type \mid epoch \mid id \mid pos]$$

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 if 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 re-broadcasts 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.

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 re-broadcasts 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

Input:
identifier id_{self} and position $\vec{P}_{self}$ of the node executing the algorithm
parameters $\vec{O}_{self}$ and d of the cluster of the node executing the algorithm
current reference point $\vec{R}_{self}$ of the cluster and epoch number e_{now}
running time T of the algorithm
Output:
identifier id_{aggr} and position $\vec{P}_{aggr}$ of the elected aggregator node

set $id_{aggr} = id_{self}$;
set $\vec{P}_{aggr} = \vec{P}_{self}$;
set timer $t_0 = T$;
set timer $t_1 = f(D(\vec{P}_{self}, \vec{R}_{self}))$;
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_{now} | id_{self} | $\vec{P}_{self}$] with max power;
  case an announcement m = [announcement | e | id | $\vec{P}$] is received:
    if the pair (e, id) has been seen before **then** drop m;
    else if $e \neq e_{now}$ **or** $\vec{P} \notin square(\vec{O}_{self}, d)$ **then** drop m;
    else if $D(\vec{P}, \vec{R}_{self}) > D(\vec{P}_{aggr}, \vec{R}_{self})$ **then** drop m;
    else
      set $id_{aggr} = id$;
      set $\vec{P}_{aggr} = \vec{P}$;
      if timer t_1 is still active **then** cancel timer t_1;
      re-broadcast m with max power;
end while
output id_{aggr}, $\vec{P}_{aggr}$

Figure 4. The pseudo-code of the aggregator election procedure of PANEL

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 pseudo-code of the aggregator election algorithm is given in Figure 4.

3.4. 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 inter-cluster routing protocol.

The intra-cluster routing protocol is used to route a message to the aggregator of a given cluster if that messages is already inside the cluster. This concerns, on the one hand, the messages that contain the sensor readings 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 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.

An important observation is that the aggregator election procedure described in Subsection 3.3 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 lifetime of the epoch in order to support the routing of queries that are destined to aggregators of past epochs.

The inter-cluster 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 inter-cluster routing protocol for two reasons: First, PANEL already makes the assumption that the nodes are aware of their positions, and therefore, this position information can naturally be re-used for routing purposes. Second, inter-cluster 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 inter-cluster routing in PANEL. Apart from being a position-based routing protocol, we do not restrict the choice for inter-cluster routing in PANEL.

The inter-cluster 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 inter-cluster 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.

3.5. Discussion

We complete the description of PANEL in this subsection by discussing three issues related to its operation: the accuracy of the node's position information, the problem of node depletion, and the security of PANEL.

Accuracy of the position information: 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 with virtual positions invented by the nodes themselves once and forever at the beginning of the operation of the network.

Besides the aggregator node election procedure, another component of PANEL, the inter-cluster routing protocol also uses the position information of the nodes. Therefore, the required accuracy of the position information is determined by the position-based inter-cluster routing protocol used in PANEL. Note, however, that one may consider replacing the position-based inter-cluster routing protocol with a

non-position-based protocol, in order to further decrease the dependency of PANEL on the accuracy of the positioning mechanism.

Depletion of nodes: 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 larger the clusters are, the more likely is that the subnetworks of the clusters will be connected given a particular node density. 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 during the aggregator election phase. Our current work is concerned with the design of an energy efficient solution to the node depletion problem of PANEL; we will report on our results in upcoming publications.

Security: 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 aggergators. In PANEL, such an attack can be perpetrated by using fake position information in the announcement message during the aggregator election phase. In particular, the adversary can invent a new node identifier and report the invented identifier very close to the current reference point in each epoch.

PANEL can be easily extended with security measures to prevent these misdeeds. First of all, announcement messages can be authenticated with a broadcast authentication scheme such as TESLA [10] in order to detect the use of invented identifiers. Second, in PANEL, the nodes keep in their routing tables the position information of the other nodes form which they have already heard an announcement. Moreover, this information is kept in the routing tables beyond the lifetime of an epoch. Therefore, the nodes can detect if a corrupted node tries to report itself at different positions in different epochs. Such behavior is deemed suspicious in a static sensor network, and can be reported to the base station.

4. Simulations

In this section, we study the energy efficiency of PANEL by means of simulations written in Matlab. In particular, we compare the lifetime of a network using PANEL as the aggregator node election protocol to the lifetime of the same network when the aggregator election protocol is LEACH [6]. Although LEACH is not a position-based protocol, we have chosen it for comparison, because LEACH also elects the aggregators in a random manner, somewhat similar to PANEL. In addition, LEACH is a well-understood, and frequently referenced aggregator election protocol.

In order to be able to compare PANEL to LEACH, we consider the same setting for the two algorithms: First of all, we perform the simulations with a static base station that resides outside the deployment area (even though PANEL is able to handle a mobile and intermittently available base station). Second, we consider that messages are sent by the aggregators to the base station in a single hop (even though PANEL supports more sophisticated query/response mechanisms).

In PANEL, the number of aggregator nodes is equal to the number of clusters which is controlled by the end-user of the network. In our simulations of PANEL, we divide the deployment area into 4 clusters of equal size. In LEACH, no explicit clustering is needed, and the number of aggregators is recommended to be around 5% of the total number of nodes.

We measure the lifetime of the network in epochs, where each epoch consists of the aggregator node election phase and a few message transmissions to the aggregator node (namely, each node transmits 5, 10, or 25 messages, depending on the simulation setting). In LEACH, the lifetime of the network is defined as the number of epochs until a given percentage of the nodes become depleted (namely 1, 50, or 100%, depending on the simulation setting). In PANEL, the lifetime is defined in a slightly different manner due to PANEL's requirement that the nodes must remain connected within each cluster. Hence, in PANEL, we

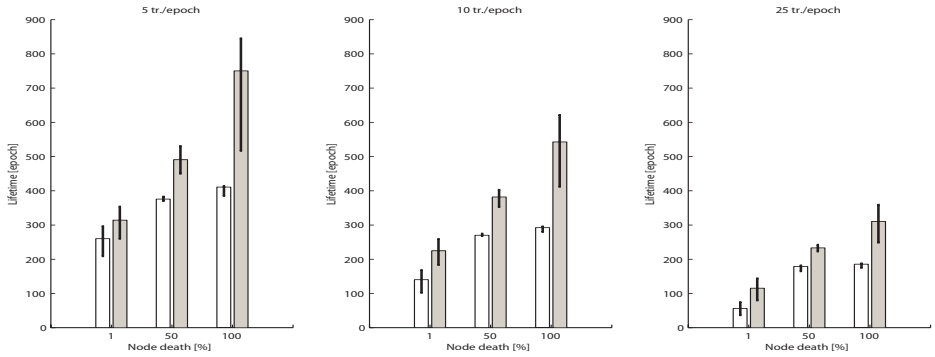

Figure 5. Lifetime comparison of LEACH and PANEL for different number of transmissions per epoch.

consider the network dead when a given percentage of the nodes are depleted *or* when the nodes of any of the clusters become disconnected (within their cluster). In the case when a disconnection occurs, we observe what percentage of the nodes were depleted, and we run the simulation of LEACH with the same settings until the same percentage of nodes are depleted.

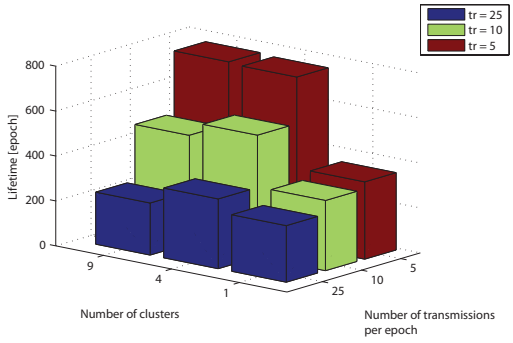

Figure 6. Lifetime comparison of PANEL for different number of clusters and different number of transmissions per epoch.

The results of the comparison are shown in Figure 5, where the white bars belong to LEACH and the gray bars belong to PANEL. The whiskers on each bar indicate the 95% confidence interval of the measured lifetimes in 50 simulation runs. As one can see, the average lifetime of the networks that use PANEL as aggregator node election protocol is always higher than the lifetime of the networks that use LEACH. Moreover, with longer simulation runs (i.e., higher percentage of depleted nodes) the difference in lifetime grows to about 67% for the case of 25 transmissions per epoch, and above 83% when fewer transmissions are performed in one epoch.

In Figure 6, the lifetime of the network using PANEL is shown for different number of clusters (i.e., different number of aggregator nodes) and for different number of transmissions in one epoch.

We can observe that the lifetime of the network increases as the number of transmissions per epoch decreases, which is consistent to our expectations. A somewhat less intuitive observation is that increasing the number of clusters decreases the lifetime of the network for higher number of transmissions per epoch. The reason is that increasing the transmission rate and increasing the number of clusters both increase the probability that the nodes of one of the clusters become disconnected (as the higher the transmission rate is, the larger the energy consumption of the nodes is, and the more clusters are in the network, the higher is the chances that at least one of them becomes disconnected). Therefore, the probability that the network dies earlier increases.

5. Related work

One of the most well-known approach for aggregator node election is the LEACH protocol [6]. In LEACH, the clustering is based on random numbers: each node picks a random number and according to its value the node becomes a cluster-head (and in the same time aggregator) or remains cluster member. The cluster members join the cluster of the cluster-head with the highest energy advertisement. The advantage of LEACH is that it flatly balances the energy consumption of the network, but it uses one-hop communication between the cluster members and the elected cluster head, as well as between the cluster heads and the base station, which can waste energy.

Other clustering protocols in the literature can be classified on the basis of how they elect the aggregator nodes. For example, in [13], the communication cost and the remaining energy of the sensor nodes is considered, while in [3], a generalized *weight* is used for this purpose. Graph theoretical approaches can be found in [2] and in [8]. In [1], the authors propose heuristics to form clusters of nodes that are within d hops away from each other, while in [4], new clusters are created as the size of the overlapping areas of existing clusters becomes small. The SANE protocol [11] combines three random aggregator node election schemes while considering adversarial attacks.

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. PANEL can guarantee a minimum distance between aggregators, because in PANEL, the aggregator nodes reside within fixed size clusters. For instance, the minimum distance between two aggregators belonging to non-neighboring clusters is dx, where x is the number of clusters between the two aggregators, and d is the cluster size.

6. 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 inter-cluster 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 energy efficiency by means of simulations. In particular, we compared the network lifetime obtained using PANEL to that obtained using LEACH, an aggregator node election protocol well-known from the literature. Our results show that the network lifetime is longer when PANEL is used.

References

[1] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh. Max-min D-Cluster formation in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, March 2000.

[2] S. Banerjee and S. Khuller. A clustering scheme for hierarchical conrol in multi-hop wireless networks. In *Proceedings of IEEE INFOCOM*, April 2001.

[3] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, 1999.

[4] H. Chan and A. Perrig. ACE: an emergent algorithm for highly uniform cluster formation. In *Proceedings of the First European Workshop on Sensor Networks*, January 2004.

[5] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. TinyPEDS: tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Elsevier Ad Hoc Networks*, June 2006.

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2000.

[7] B. Karp and H. T. Kung. Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM Mobicom*, August 2000.

[8] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proceedings of ACM Mobicom*, September 2004.

[9] M. Maróti, B. Kusy, , G. Balogh, P. Völgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi. Radio interferometric geolocation. In *Proceedings of ACM SenSys*, November 2005.

[10] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

[11] M. Sirivianos, D. Westhoff, F. Armknecht, and J. Girao. Non-manipulable aggregator node election protocols for wireless sensor networks. In *Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2007.

[12] S. Čapkun, M. Hamdi, and J. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Cluster Computing Journal*, **5**(2), April 2002.

[13] O. Younis and S. Fahmy. Distributed clustering in ad hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of IEEE INFOCOM*, March 2004.