Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2009

Project Title: Opinion Formation in Sensor Networks
Student: A. Kleerekoper
Course: ISE4
Project Supervisor: Dr J. V. Pitt
Second Marker: Professor E. Gelenbe
Abstract

Sensor networks suffer from low energy which limits their usefulness for real applications. In this paper we look at the accuracy of these networks and some energy saving techniques. We propose and develop an aggregation algorithm based on opinion formation models originally designed to model human interactions. This algorithm allows the adaptive trading off of accuracy for longevity and vice versa. Simulation results show that the algorithm increases accuracy by up to 50% and allows this accuracy to be traded for longevity to increase the life-span of the network by almost three times. We present some of the possibilities for future work that has been opened up by this innovation.
Acknowledgements

I would like to thank Dr Jeremy Pitt for his help and guidance throughout the project. He was available at all time to offer assistance and our regular meetings helped keep the project progressing smoothly. I would also like to thank Brendan Neville for his time and effort developing Presage and helping me learn to use it effectively. Thanks to Dr Gurcan for giving up his time to teach me. Final thanks to David Blancke for collaboration and acting as a sound board for my ideas.
Contents

1 Introduction ................................................................. 4

2 Background ................................................................. 7
  2.1 Application Classes .................................................. 7
  2.2 Energy Saving Algorithms ........................................... 9
  2.3 Self Organisation ..................................................... 12
  2.4 Aggregation ............................................................ 15
  2.5 Sensor Networks as Multi-Agent Systems ......................... 16
  2.6 Opinion Formation ................................................... 17

3 Problem Definition ........................................................ 19
  3.1 Formal Definition ...................................................... 19

4 Proposed Solution ........................................................ 21
  4.1 Assumptions ........................................................... 21
  4.2 Routing and Self Organisation ....................................... 23
  4.3 Aggregation ............................................................ 27
  4.4 Intelligent Broadcasting and Network Reformation ............... 30
5 Simulation Platform and Procedure .................................. 32
6 Simulation Results ......................................................... 35
7 Conclusion ............................................................... 41
8 Future Work ............................................................. 41

A Power ................................................................. 43


1 Introduction

Sensor networks are ad hoc, wireless networks comprised of hundreds, thousands or even hundreds of thousands of sensor nodes and one or more sink nodes acting as gateways to the end user. They have become a reality thanks to recent advances in micro-electronics that have allowed the creation of low-cost sensor nodes containing on-board sensors, a microprocessor and a transceiver.

Applications for sensor networks are varied and include detection of biological, chemical or nuclear attacks [26] and habitat monitoring [30]. [6] charts the short history to date of the development of sensor networks and outlines many potential applications. [2] offers a survey of active research in the field.

The primary advantage of sensor networks over more traditional sensor arrangements is that their low price allows for far more sensors to be deployed for the same cost as a small number of alternative sensors. In fact [2] claims that sensor nodes ‘should cost much less than $1 each in order for the sensor network to be feasible’. Having more sensors results in three advantages. The first is that the exact location of a phenomenon does not need to be known in advance - they provide greater area coverage. The second advantage is that more readings are taken which can help improve accuracy, although as we shall see this is not necessarily the case. The final advantage is that the network is robust to the failure of individual nodes.

Logistically, the large number of nodes in a sensor network poses a major problem, yet the solution to that problem brings new benefits. It is not feasible to plan the precise locations of the sensor nodes in advance of deployment and design a network topology for them. Therefore, sensor networks must be capable of self organisation to produce a method of routing messages from sensor nodes to the sink node(s). If such self organisation can be efficiently implemented, and much research has been and is being carried out on this point, the network becomes highly scalable and easily deployable. Self organisation provides increased autonomy and allows a network to be deployed into an area that is inaccessible or even hostile to human activity. It also allows for the network to be extended by simply introducing new nodes into the area of interest and triggering the network to self organise.

Along with their size, the main characteristic of sensor networks is that they are extremely power-constrained. Sensor nodes are powered by typically low-capacity batteries.
with no recharging capability. As a result a lot of research has gone into improving the energy efficiency of sensor networks. Most power used by sensor networks is taken up by the transceiver and therefore focus is on reducing the number of messages or the energy cost of sending them.

Among the many algorithms that have been proposed, in-network data aggregation is very popular. [18] suggests that it may be the most energy-efficient class of algorithms developed so far. In-network aggregation reduces the number of transmissions being sent by the network by collating information en route to the sink node. Optimally this would reduce the number of transmissions “per round” to just one per node. While aggregation reduces the number of transmissions and saves energy it also results in a loss of accuracy.

The sensors in the network produce unreliable data [39]. There are a number of factors affecting this. One is that the micro-electronic sensors are simply not as accurate as more expensive sensors can be. For example, the on-board thermistor supplied with the Mica2Dot is the Panasonic ERT-J1VR103J which has an accuracy of ±2°C [3]. This is some three orders of magnitude less accurate than the Isotech TTI-22 which has an accuracy of ±0.001°C [15]. Another reason why sensor nodes provide inaccurate readings is that they have not been carefully placed in specific locations. Their random deployment could well lead to individual nodes being situated in areas that skew their readings, e.g. a temperature sensor located next to an air conditioning unit or heater. Other causes include noise and sensor malfunction [39].

Averaging the errors should compensate for the presence of noise and remove the impact of individual erroneous readings. However, if aggregation is carried out in-network, en route to the sink node then the number of readings being averaged will be relatively small allowing anomalous readings to have a disproportionate effect on the averaged result and thereby reducing the accuracy of the entire network. In-network aggregation (hereafter referred to simply as aggregation) therefore provides an energy - accuracy trade off. We can aggregate to save energy and lose accuracy or else not aggregate and gain accuracy at the cost of the longevity of the network.

In this project we aim to propose a new aggregating algorithm that can be used to adaptively trade off the accuracy of the network for longevity and vice versa. We hope to show that a network can be given an error margin to stay within and change its accuracy to increase its longevity while remaining within that margin. We create a simulation of a sensor network in order to test our algorithm.
Before formally defining the problem, in section 3, and outlining our proposed solution, in section 4, we first present some background information and discussion of past work carried out in relevant areas. In section 5 we explain the platform and procedure used for simulating the algorithm and in the following section (6) we present the results of our simulations. Finally, we conclude and offer some suggestions for future work.
2 Background

2.1 Application Classes

To date numerous potential and real applications have been suggested for sensor networks. Many examples of both can be found in [38] and [25]. From the literature we have seen, categorisation of applications tends to be based on the end-user’s requirements rather than any fundamental characteristic of the network itself. As an example, the call for papers for the 1st ICST International Conference on Sensor Networks Applications, Experimentation and Logistics (SENSAPPEAL 2009) defines application classes as environmental monitoring, physical security and surveillance, health care etc [27]. In our view, though, key network characteristics depend more on the nature of the data gathering process than on the purpose of the data being gathered.

We therefore define four application classes - Continuous Tracking, Event Monitoring, Continuous Aggregation and Triggered Aggregation. These classifications rely on the interplay of two requirements, which we call periodicity and source identification. Periodicity refers to the network gathering data at regular intervals as opposed to gathering data only after some triggering event. Source identification refers to the question of whether the location or identity of the sensor node that produced the data is required along with the data or whether the data alone is requested. The definitions of the four classes based on the requirements is given in table 1, below.

<table>
<thead>
<tr>
<th>Class</th>
<th>Periodicity</th>
<th>Source Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Tracking</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Event Monitoring</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Continuous Aggregation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Triggered Aggregation</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 1: Table showing the definition of application classes

Continuous tracking applications are ones in which data is required periodically but the source of that data must be known as well. An example would be measuring the concentration of a contaminant in a water supply at certain locations. Event monitoring applications also require the source of the data to be known but do not produce periodic data. Instead they produce data when certain triggering events occur. An example of this
class would be monitoring the concentration of a contaminant in order to alert the end user if the concentration in a location rises above a certain threshold. Continuous aggregation applications produce data periodically but the source of the data is not required. An example would be a network designed to inform the end user of the average concentration of the contaminant. Triggered aggregation applications produce data only in response to some triggering event and the source of individual data items is not required. An example would be informing the end user if the average concentration rises above some threshold.

All the examples above would have been categorised simply as ‘environmental monitoring’ in the literature but the different requirements present different problems and impose different limitations on what algorithms can be suitably applied to them. We now go on to discuss some of the energy saving algorithms proposed for sensor networks and show how they are applicable to the different classes of application we have defined.
2.2 Energy Saving Algorithms

Sensor nodes are extremely power-constrained because they generally only have small capacity batteries with no possibility for recharging. As a result, the usefulness of sensor networks is limited by a short life span. Much research has been carried out into different techniques for improving the energy efficiency of the network in order to extend their life-span. The methods can be loosely categorised into three groups. The first group reduces energy cost by reducing the number of messages being sent around the network. It does this by putting some nodes into a sleep mode. The second attempts to reduce the cost of transmitting individual messages being sent from one node to another. The third reduces the number of messages by joining messages together en route to their destination.

An example of the first type is the Randomised Independent Sleeping (RIS) scheme outlined in [19]. In that scheme time is divided into periods and at the start of each period nodes choose whether to enter a sleep mode or an active mode at random. The probability of moving into an active mode is $p$ and of going to sleep is $1-p$ thus theoretically extending the life of the network by $\frac{1}{p}$. They point out that finding an appropriate value of $p$ is trivial given the lifetime of individual nodes and the required lifetime of the network.

[35] explores the assumptions underlying various sleep-based energy saving schemes. One of the fundamental assumptions in RIS is that every sensor node in the network is a single hop from the sink node. If individual nodes are multiple hops away then in order to be useful to the network not only must they be active but there must also exist a path of active nodes between themselves and the sink. The assumption of a single hop network means that most sleeping schemes are unsuitable to large sensor networks.

[40] proposes a sleeping scheme for multi-hop networks which they call S-MAC. While RIS reduces the number of messages in the network by having some nodes move to a sleep mode and thereby not produce messages, S-MAC uses sleep modes to reduce the energy wasted in having a radio receiver idling. To do this, nodes constantly switch between sleep modes in which their receivers are off and active modes when they are on. Messages can only be transmitted during active modes and neighbouring nodes synchronise so that active modes overlap. They recognise that message throughput will be reduced and latency would occur and devise a scheme to compensate but this results in their sleeping scheme only providing energy savings when traffic is low.

Since RIS has nodes sleeping for some time periods during which they do not transmit
their readings it is clearly not suitable for applications requiring periodic reports. However, for triggered events, sleeping schemes like RIS could be very effective but they would be limited to small networks. S-MAC, on the other hand, is a sleeping scheme that could be used for all application classes as it makes no assumptions about the nature of the network. Nevertheless, since it is only effective when network traffic is low it would seem to be more suited to those aggregating applications in which the number of messages is reduced.

[2] presents some options for reducing the amount of energy required to send data from one node to another. Clearly, the amount of energy required for a single hop depends on the distance between the two nodes so energy saving relies on finding energy efficient routes between distant nodes. One option is called Maximum Available Power in which nodes attempt to send messages along routes whose nodes collectively have the most energy left. This method would lead to more balancing of the energy requirements so that routes whose nodes have less energy will be avoided in favour of those with more energy and thus networks should last longer before routes start to close completely. Obviously care must be taken not to include super-sets as valid options.

A slight variation on the first method is Maximum Minimum Power Available. This takes into account not just the energy of the route but also of individual nodes within a route and effectively lowers the energy available of a route to the energy available in the weakest link. Another method is Minimum Energy which totals the energy required to transmit along routes and thereby pays attention to the distances being covered. A similar option is Minimum Hop which routes messages along the shortest topological path. These two are identical when hops all require uniform energy.

While these methods reduce the energy required to send messages over a large distance through which numerous routes exist they become less effective when bottlenecks exist. This is likely to happen when all data is designed to reach a sink node as only a few other nodes are going to be a single hop away leaving only a small number of alternative routes. They also include the overhead of forming and maintaining accurate routing tables. This general approach to energy efficiency is best suited for applications where source identification is required. This is because these methods do not interfere with the messages themselves and thus do not remove information from them about the sender.

The final type of energy saving algorithm is the in-network data aggregation mentioned above. [18] claims that this method saves 50% to 80% more energy than efficient routing. Data aggregation reduces the number of messages by combining the data being sent with
data included in other messages flowing along the same route. Aggregation can take one of two forms. One form is where the resultant message is a condensation of the other messages and is itself the same size as the messages from which it was formed. The other form is a concatenation of incoming messages. The first form provides greater savings as the amount of time during which transceivers are in use is reduced significantly while the second form does reduce total transmission times a little bit it mostly makes improvements from increased efficiency of medium access.

The two forms of data aggregation lend themselves to different aggregating application classes. Where the aggregation function produces an outgoing message of the same size as an incoming one (eg max, min, average) the method is best suited for aggregating applications where source identification is not required. Where source identification is required the only form of data aggregation that is suitable is concatenation of some sort.
2.3 Self Organisation

The ability of a sensor network to organise itself is critical as the very large number of nodes precludes careful placement in a designed topology. However, self organisation also impacts the energy efficiency of the network as we shall now show.

![Diagram showing the connection between nodes in a network using flooding](image)

Figure 1: Diagram showing the connection between nodes in a network using flooding

Self organisation can be achieved very easily by flooding. As shown in figure 1, flooding involves nodes sending messages to all their neighbouring nodes. Upon receipt of a message, nodes simply relay it to all their neighbouring nodes unless the message was meant for them (or originated from them). While this organisation is incredibly simple it is grossly inefficient. [13] identifies some of the problems with it. The first is Implosion which is the result of a message being duplicated when it is first broadcast and then the multiple copies being relayed to a single node. Although not mentioned in [13] the corollary of Implosion is Explosion where duplicates are created again and again mostly in the opposite direction to the destination. They do cite problems of overlapping
data when the same information is sent by more than one node and they point out that flooding is unaware of the resources available and makes no attempt to route messages efficiently. Clearly flooding is the least energy efficient method as it creates numerous messages and floods the duplicates throughout the entire network.

The topology created by flooding can be made more efficient with some of the methods mentioned earlier for selecting power efficient routes. Applying the minimum energy method to the network produces the routes shown in figure 2. The diagram shows why this method is not best suited for aggregating applications as nodes are often relaying messages from only a small number of other nodes and therefore aggregation produces little benefit.

Figure 2: Diagram showing the connection between nodes in a network using minimum energy routing

[12] proposes a self organisation algorithm that produces a clustered topology. Low-Energy Adaptive Clustering Hierarchy (LEACH), shown below in figure 3, is designed for networks in which every node is a single hop from the sink node and assumes that nodes are capable of changing the range of their transmissions. Their algorithm has every node randomly deciding when to declare itself a clusterhead with some probability $p$. When a
node declares itself a clusterhead it offers its services to neighbouring nodes. Any nodes in the vicinity that are not themselves clusterheads at that time choose between competing clusterheads based on the amount of energy required to reach them. As a result nodes are formed into clusters in which non-clusterheads are one short hop from the clusterhead and the clusterhead is one hop from the sink. The aim is to share the high energy cost of transmitting to the sink node. LEACH was designed for data aggregation and is well suited for it as the clusterheads provide convenient points to aggregate data.

However, LEACH is not without its disadvantages. For one thing the requirement that all nodes are one hop from the sink limits LEACH to small networks. A bigger drawback, though, is that, as the diagram shows, some data is being transmitted away from its destination before being transmitted to the sink. The total distance covered by transmissions is therefore sub-optimal.

Figure 3: Diagram showing the connection between nodes in a network using LEACH
2.4 Aggregation

Aggregation in sensor networks has been proposed for a number of reasons. One, as we mentioned earlier, is that it reduces the number of messages being sent around the network and therefore saves energy. Others include making the incoming data stream more manageable and reducing the time needed for further computation, [29]. The result of some aggregation functions are unaffected by in-network aggregation but for others aggregation reduces the accuracy of the results.

[39] makes the point that sensor nodes produce unreliable data. This is partly because they are less accurate than larger, more expensive sensors. For example the on-board thermistor supplied with the Mica2Dot is the Panasonic ERT-J1VR103J thermistor which has an accuracy of ±2°C [3]. This is some three orders of magnitude less accurate than the Isotech TTI-22 which has an accuracy of ±0.001°C [15]. It is also because of environmental noise, malfunctions and, because nodes cannot be carefully placed, poor location could affect the data, for example a temperature sensor next to an air conditioning unit.

Functions like max and min are unaffected by in-network aggregation but average would be. The impact of erroneous readings is lessened when the average is taken over more readings. Since individual readings are inaccurate to begin with, in-network data aggregation, while saving power, reduces the accuracy of the final result. This trade off is noted by [4] who proposes an algorithm for estimating global max that can save energy at the cost of accuracy of the estimate or improve accuracy at the cost of extra energy. Their approach is not applicable to summation aggregations such as average.
2.5 Sensor Networks as Multi-Agent Systems

[37] defines an agent as:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

and a multi-agent system as:

A multiagent system is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure.

Using those definitions it is easy to see why sensor networks are being viewed as multi-agent systems, as in [34] and [33]. Our discussion of sensor nodes show that they are required to act autonomously in order to self organise and be able to operate unattended in inaccessible or hostile locations. Sensor nodes are placed in some environment in order to interact with that environment to perform some sensing task. If we view a sensor node as an agent we have autonomy in place and can say that its sensing and relaying of data is autonomous action in its environment to meet its design objectives.

The definition given of a multi-agent system translates exactly into a sensor network once we view the nodes as agents. The network is a system that consists of a number of agents (nodes) which interact by exchanging messages through a wireless network.
2.6 Opinion Formation

Multi-agent systems have been used to model the way in which people form opinion on topics. We briefly present some existing opinion formation models.

[32] presents a model for binary decisions where agents have a choice between choosing option A or option B. Their model revolves around the interaction between neighbouring agents as shown in figure 4. When agents $d$ and $e$ agree with each other then their six neighbouring agents adopt the same opinion. When $d$ and $e$ disagree the neighbouring agents also disagree, thus $a$, $c$ and $g$ disagree with $d$ while $b$, $f$ and $h$ disagree with $e$. Their model will always result in one of two stable results, either all agents adopt the same opinion or 50% adopt one opinion and 50% adopt another with every agent disagreeing with each of its neighbours. The authors offered no justification for the correctness of their model.

![Figure 4: When agents d and e are chosen to interact their opinions effect the opinions of their six neighbours](image)

[8] propose a model that deals with opinions that are continuous on some topic. An opinion, in their model, can be any real number in the range $[0..1]$. In their model two randomly chosen agents meet. If their opinions are similar to each other then they share opinions. Similarity is given simply by the absolute difference of opinion being less than some threshold known as the confidence bound. Sharing depends on some variable and the difference in opinion. After agents $i$ and $j$ meet, if the difference is below the threshold, the new opinion for agent $i$, $x_i$, is given by the following equation involving some variable $\mu$ used to control the sharing of opinions:

$$x_i = x_i + \mu(x_j - x_i)$$

The justification for the model is that people only discuss their opinions when some common ground exists. If their starting points are too divergent then neither is capable
of influencing the opinion of the other. Their simulation results show that under this model agents converge to a common opinion if the values for the confidence bound and \( \mu \) are chosen correctly. For different values, two distinct opinions emerge and all agents converge to one of those opinions. In this model confidence is a factor of the opinions not of the agents. This seems a little unrealistic as the person who holds the opinion, at least to some extent, has an effect on the sharing of that opinion. Some people are better orators than others or are generally held in higher esteem and their opinions are more influential even on people who started off with a vastly different opinion.

[24] presents a model for continuous opinions in which all agents interact with each other and share opinions. They propose a model of dynamic confidence in which the impact of an agent’s opinion is a function not only of the opinion itself but also of the confidence each other agent has in it. Confidence in their model is based on some mind-set and how well the opinion expressed by an agent conforms to some pre-existing mind-set. This seems more accurate as confidence depends on the individuals rather than the opinions and there is sharing even from agents with vastly different opinions although their confidence is likely to be low leading to little impact. The model still doesn’t provide a measure for how susceptible an agent is to the opinion of others. There are some people whose opinions are fixed regardless of the opinion of others and some who change their opinion extremely easily.
3 Problem Definition

We have seen that sensor networks are extremely power-constrained and hence have relatively short life-spans which limits the usefulness of the networks. We have also seen that the readings produced by individual sensor nodes is unreliable because of their reduced accuracy and the increased possibility of poor placement. We have noted that in-network data aggregation can increase the life-span of the network by reducing the number of messages to just one per node per round for aggregation functions like average. However, the improved longevity comes at a cost to accuracy for the averaging function as erroneous data has more of an impact when it is averaged together with a smaller number of more correct readings than it would were it transmitted to the sink to be averaged there along with the readings from every other node. A trade off therefore exists between accuracy and longevity for networks implementing aggregation functions like average.

The aim of this project is to develop an algorithm for aggregation that can replace simple averaging to produce more accurate results and at the same time allow for the adaptive trading off of that accuracy for longevity. To that end we propose the following scenario. We consider a sensor network inside a large factory designed to record the average temperature of the room. Each sensor takes a reading every time step which is incorrect by some error margin. Each node transmits its reading to some other node which stores it for some time before aggregating all the readings it has received, including its own reading, to form some opinion of the correct reading. This is then transmitted to yet another node and so on until the sink node forms an opinion. The network reading is the opinion formed at the sink node for a given time and the network error is the absolute difference between the network reading and the correct value. The network lifetime is the number of time steps in which nodes besides the sink have energy left.

The problem is to derive an algorithm for aggregation that allows us to specify an upper accuracy bound for the network error such that the network lifetime increases as the upper bound is raised.

3.1 Formal Definition

- We have a set of sensor nodes $S = \{s_1, s_2, s_3, ..., s_n, s_{\text{sink}}\}$ where $s_{\text{sink}}$ refers to the sink node
• Each sensor node, $s_i$, has some energy at time $t$, $en_i(t)$, and some, fixed, error value, $er_i$

• Sensor nodes take readings such that $r_i(t) = R(t) \pm er_i$ where $R(t)$ is the correct value. The error value is applied positively and negatively at random

• Sensor nodes form opinions such that $o_i(t) = O(\{r_1(t), r_2(t), ..., r_n(t)\})$ where $O(\{r_1(t), r_2(t), ..., r_n(t)\})$ is some aggregating function of the readings received from nodes $1, 2, ..., n$

• Network error is defined as $NE(t) = abs(R(t) - o_{sink}(t))$

• Network lifetime is defined as $NL = t$ such that $\sum_{i=0}^{n} en_i(t) > 0$ and $\sum_{i=0}^{n} en_i(t + 1) \leq 0$ where the summations do not include the sink node

The problem is to derive an algorithm for $O(\{r_1(t), r_2(t), ..., r_n(t)\})$ such that we maximise $NL$ while ensuring that $\forall t[NE(t) \leq x]$ where $x$ is some predefined error bound.
4 Proposed Solution

4.1 Assumptions

Before outlining the algorithms we devised for our solution we first list and justify the assumptions we made.

- We follow [9] in assuming that the sink node has infinite energy. This is reasonable because the sink node is designed to transmit information to the base station which requires far more energy than the short range transmissions of the sensor nodes and therefore is likely to be fitted with a larger battery which will certainly outlast the batteries of the sensors. If the sink node will certainly outlast the rest of the nodes in the network then we can ignore its energy and consider it infinite.

- We assume that the nodes are able to transmit messages at different strength using different amounts of energy. This is justified on the basis that the Chipcon CC1000 transceiver used by the Mica2Dot sensor node is capable of different transmission strengths.

- We assume that when comparing energy consumption in the network we need only consider the energy used in transmitting messages. This is because our algorithms are unlikely to significantly increase the time spent on computation to a level that would rival the power drain from the transceiver. As far as the transceiver is concerned, the power required for receiving messages is about the same as that required for idling and since this is constant our algorithm would not affect it. All our algorithms affect is that number of messages being sent and so therefore for comparison purposes we need only consider the energy drain from that.

- We assume that error is relatively constant with respect to time. Sources of random error in temperature sensors include thermal drift, changes in readings caused by external factors such as a heater that is suddenly switched on and large changes in temperature affecting the resistance of the thermistor [31]. Since our scenario is unlikely to include large changes in temperature and since thermal drift is likely to affect all nodes to the same extent we can discount random errors and focus on fixed ones.

- We assume that all nodes are synchronised to one clock. While this is not feasible in reality it is reasonable to assume it if we divide time into small periods. While
clocks are not going to be synchronised perfectly it is reasonable to assume that it is possible to synchronise them to approximately the same time enough to operate in time periods that are closely synchronous.

- We assume that no collisions take place between messages. This is reasonable as messaging takes place inside a time step period and clusterheads are in a position to schedule when messages should take place using one of the many schemes already devised to avoid collisions.

- We assume that no messages are lost. This has been made for simplicity but cannot be said to hold in reality. However, we feel that the impact of lost messages on the effectiveness of our proposed solution is so minimal that it can be safely ignored.

- We assume that nodes have a globally unique ID. While this cannot be guaranteed, all that is actually required is that two nodes with the same clusterhead do not have the same ID. If a random number is generated as the ID of a node within a suitably large range the probability of a clash is so extremely small as to be negligible.
4.2 Routing and Self Organisation

We propose using an aggregation tree for routing readings to the sink node. The tree topology is essentially a clustering protocol and therefore easily lends itself to data aggregation as clusterheads can aggregate the data received from their clusters. We avoid LEACH because we do not assume that all nodes are a single hop from the sink. A protocol for forming an aggregation tree is presented in [9] and we follow that with minor adjustments. Their protocol uses just a single message. That message contains the ID, parent and power of the sender as well as the number of hops from the sender to the sink. They suggest that each node have a timer that starts counting down upon reception of the clusterhead request message. The timer length is the reciprocal of the energy left in the node and thus a node with more energy will finish its countdown quicker. When a countdown is completed nodes choose a parent from among those nodes that sent them a request based on the residual energy of that node and the hop count to the sink. It then broadcasts its own request identifying its new parent in the message. Its parent would also receive this message and would conclude that it is the parent of the sender.

One of their underlying assumptions is that all sensors have the same transmission range. This leads to their usage of the request itself also as a confirmation of acceptance. As mentioned above we have assumed that the transmission range is variable. This difference is important as it leads them to ignore the transmission cost to the clusterhead as a criterion for choosing between offers. If transmission range is variable then it is evident that that plays a greater role in choosing a head than the energy of the clusterhead itself. This is because the energy left in the clusterhead affects only the ability of that node to send on information whereas the distance between the two nodes affects the power required in every transmission from both nodes.

We also believe that, while they have designed this protocol with aggregation in mind, they may not have fully appreciated the implications of that decision. If aggregation is to be employed then the number of transmissions in the network should approach one message of equal size per node per round, including clusterheads. As such it is of significantly less importance how much energy a potential clusterhead has. Further, we believe that the number of hops between a potential clusterhead and the sink is irrelevant if aggregation is taking place. This is because the aggregation latency is set to the delay of the longest route and for all other nodes there is no benefit derived from having their reading reach the sink any earlier. This is even more so when the readings from individual sensors are to be aggregated en-route because clusterheads would most likely be waiting.
for readings from all their children before aggregating and relaying.

We therefore propose that the criterion upon which to choose a clusterhead is the transmission range required to reach it. Our clusterhead request message contains the ID of the sender and the strength it used to send the message. It also contains other information which is not related to routing but which we will discuss later in section 4.4. Nodes choose their clusterhead from among competing offers based on the strength that is required to transmit to that head.

Their concern with energy led them to implement an energy-related timer in order to have the most energetic nodes act as clusterheads by counting down quicker and thus offering before others do. We believe that this would lead to uneven cluster sizes as nodes with more energy bring nodes into their cluster that are closer to other nodes that have slightly less energy. From an aggregation point of view balanced clusters provide a more even aggregation and are better suited to our proposed method of aggregation. Because we feel that transmission range is the most important criterion and also want to have approximately equal cluster sizes we propose the following method for forming a network.

The sink node starts broadcasting an offer at some lower-bound range. It waits for explicit acceptance of the request. Acceptance must be explicit because the transmission range of an offer may be less than the transmission range required to reach a clusterhead and therefore the acknowledgement method used in [9] won’t be reliable in our case as the clusterhead may never receive a clusterhead request message from its children. The sink node counts the number of nodes in its cluster and compares it to a pre-set cluster size. If it has too few nodes in its cluster then it increases the range of its transmission, retransmits the request and again awaits acceptance messages. This process continues until either the cluster is the requisite size or else some upper-bound on transmission range has been reached. While this method involves transmitting the same message to some nodes more than once this cost is rarely occurring and reduces the risk of differing cluster sizes as well as reducing the transmission range between nodes and their clusterheads which has a longer-lasting impact. The sink node must also keep track of the strength at which it received replies so as to avoid transmitting all future messages further than the furthest node in the cluster.

Once the sink has formed its cluster it explicitly orders the nodes in that cluster to form their own. The command also contains the required cluster size leaving it within the control of the sink to alter it. By synchronising this order, approximately even cluster sizes can be formed. In an ideal situation this method would produce a perfect
aggregation tree in which every cluster is identically sized and in which all nodes serve both as cluster and clusterhead save for the nodes at the extremity of the sensor field. An ideal aggregation tree is shown in figure 5.

[Image: Diagram showing the connection between nodes in a network using our proposed aggregation tree algorithm]

Figure 5: Diagram showing the connection between nodes in a network using our proposed aggregation tree algorithm

[16] makes the point that knowing how long to wait for messages is an important question for networks organised as aggregation trees. Waiting until all nodes in a cluster have sent data can lead to a blockage in the tree if a node suddenly loses power or is incapable of sending a message. Even providing a long time-out can cause delays which give rise to latency issues and stale data. We propose that nodes be given knowledge of what "level" they are at within the tree. We achieve this by having all nodes start at level 0 signifying that it is a leaf node at the extremity of the sensor field. A node that has just formed a network must, by the nature of our method, be at level 1. It can then inform its own clusterhead to move up one level and so on up to the sink node. With this
information in hand nodes can schedule their aggregation with almost no latency increase. A node at level 0 will "aggregate" immediately and broadcast its aggregated reading to its clusterhead. It takes some time for a cluster-worth of readings to be transmitted to the clusterhead and, given approximately equal cluster sizes, it is therefore possible to assign a time-step for aggregation where a time-step is the time required for all nodes in a cluster to broadcast their aggregations to the clusterhead. As such, each node in the tree can schedule its aggregation to take place at time $t + (l \times p)$ where $t$ is the time of the reading, $l$ is the level of the node and $p$ is the period of time required for all nodes in one cluster to transmit their readings to their clusterhead. \(^1\)

\begin{algorithm}
\caption{Algorithm for Forming a Cluster}
\begin{algorithmic}
\State numberOfChildren := 0
\State transmissionStrength := initialStrength
\State usefulStrength := 0
\While {(numberOfChildren < maxNodes) and (transmissionStrength < maxStrength)}
\State broadcastMessage(ID, transmissionStrength) \textcolor{red}{@} transmissionStrength
\State wait someTime
\State numberOfChildren := NumberOfChildren + numberOfReplies
\If {numberOfReplies > 0}
\State usefulStrength := transmissionStrength
\EndIf
\State transmissionStrength := transmissionStrength + strengthIncrement
\EndWhile
\State broadcastMessage(formCluster) \textcolor{red}{@} usefulStrength
\If {clusterHeadID ! = "" and ! cluster.isEmpty}
\State broadcastMessage(levelUp) \textcolor{red}{@} toClusterHeadStrength
\EndIf
\end{algorithmic}
\end{algorithm}

\(^1\)After devising this method for scheduling aggregation we found that [29] had proposed the same method which they were calling cascading timeouts
4.3 Aggregation

As we have mentioned earlier in section 2.4, averaging readings in many aggregation applications is a means of overcoming errors to increase accuracy. Given that the unreliability of some sensors is a motivation for aggregation we propose that the aggregation algorithm itself should take this into account. We therefore propose that rather than blindly averaging readings, clusterheads should attempt to build up a picture of the accuracy of readings from particular nodes and make use of that information when aggregating. Clusterheads should have some measure of confidence in nodes.

We have also mentioned earlier in section 2.5 that sensor networks can properly be viewed as multi-agent systems. This allows us to borrow opinion formation models designed for multi-agent systems and adapt them to sensor networks. This is promising because some opinion formation models include information on confidence as part of the model, as we wish our aggregation algorithm to do.

Of the models discussed in section 2.6 the model in [24] presents the best model for our purposes. In that model agents had a confidence in other agents given by the function \( w_i(j, t) \) mapping the confidence of agent \( i \) in agent \( j \) at time \( t \). They argued that confidence was a relative measure not an absolute one and therefore adopted the rule that the sum of all confidences (including self confidence) must equal one, i.e. \( \sum_{j=0}^{n} w_i(j, t) = 1 \). They also proposed a model of a mindset which could be used to alter an agent’s confidence in another on the basis that if agent \( i \)’s opinion was in line with the mindset of agent \( j \) then agent \( j \) would have more confidence in \( i \) than in another agent whose opinions did not accord with that mindset.

We propose adapting their model to apply to our scenario. We retain the rule that the sum of confidences should equal 1. Where they used the concept of a mindset to alter confidences we use error margins. To that end we need nodes to build up some close approximation of the error in the readings of other nodes. In order to calculate the error in a node’s reading we need to find the absolute difference between the reading and what is considered the correct value. Since the actual correct value is unknown by the nodes we provide two choices. The node can either use the aggregated opinion it found or the aggregated opinion provided by feedback from its clusterhead. We assume that the aggregated opinions will become more accurate the further up the tree one goes because more readings will have been included in the aggregation. Therefore, if a feedback value is provided the nodes will use that as their correct value, if not they will use their own
value.

Averaging the error values calculated for each node would be a simple method but runs the risk of giving too much emphasis to the initial error values. This is a particular problem when the opinions formed early on are likely to be less accurate than ones formed later. Nevertheless, we don’t want anomalous results to skew the confidence in an otherwise reliable node which could happen if we took a rolling average of the last few readings. We note that the difference in average is given by the following equation:

\[ \Delta = \frac{x+y}{n+1} \]

where \( x \) is the old average, \( y \) is the new reading and \( n \) is the number of values before \( y \). Thus if we start with a large value of \( n \) anomalous readings early on will not have a large impact while consistently accurate readings later will be able to change the average. Some testing is required to find the optimal value for \( n \).

With the error value as a basis for confidence we must produce confidence values that always sum to 1. Whereas in [24] every agent knew the opinion of every other agent every round, in our scenario this is not the case. We cannot, therefore, have a confidence algorithm that adjusts the confidence in all nodes at the same time based on their opinions. This precludes the possibility of a continuously changing confidence function for every other node. We propose the use of instantaneous confidence.

Instantaneous confidence uses the error values as the basis for having a relative confidence in some subset of the nodes in the network. We observe the a node that is twice as accurate as another ought to command twice the confidence. Thus we see that the confidence in all nodes in a given subset can be reasonably calculated in terms of their error relative to the most inaccurate. We propose the following algorithm.

Given a set of nodes, first identify the node with the highest error margin. Next calculate the ratio of the error of every node (including the worst one) to the error of the worst node such that \( x_i = \frac{e_i}{e_w} \), where \( x_i \) is the ratio of errors for node \( i \) and \( e_w \) is the error of the worst node. Summing these ratios gives:

\[ \sum x = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} \frac{e_i}{e_w} \]

If the instantaneous confidence in the worst node is defined as \( C_w = \frac{1}{\sum x} \) then the con-
fidence in all other nodes can be defined as $C_i = x_i \times C_w$. The instantaneous confidences provably sums to 1 as follows:

$$\sum_{i=1}^{n} C_i = \sum_{i=1}^{n} x_i \times C_w = \Sigma x \times C_w = \Sigma x \times \frac{1}{\Sigma x} = 1$$

With these instantaneous confidence values we propose aggregating the readings as a weighted average. Thus the aggregated value formed at a clusterhead would be given by:

$$o_i(t) = \sum_{i=1}^{n} r_i(t) \times C_i(t)$$

This algorithm is self improving as clusterheads form more accurate values for the error of the nodes in its cluster. Below is the pseudocode for the algorithm for finding instantaneous confidences from error margins:

**Algorithm 2** Algorithm for Instantaneous Confidence

```
worstNode := ""
worstError := 0.0
for every node that sent a reading as Node do
    if Node.error > worstError then
        worstError := Node.error
        worstNode := Node
    end if
end for
for every node that sent a reading as Node do
    Node.ratio = Node.error / worstError
end for
ratioSum := 0.0
for every node that sent a reading as Node do
    ratioSum := ratioSum + Node.ratio
end for
worstNode.conf = 1 / ratioSum
for every node that sent a reading as Node do
    Node.conf = Node.ratio * worstNode.conf
end for
```
4.4 Intelligent Broadcasting and Network Reformation

With aggregation using error values to assign instantaneous confidences we are in a position to trade off accuracy for longevity. Some of the nodes will have large errors and the confidence in them will be low. Their readings will have a small impact on the final opinion. Little is gained by these nodes broadcasting their reading. By not broadcasting they save their energy with only a small decrease in accuracy. With this in mind we provide a framework for nodes to decide whether to broadcast their readings or not.

If the clusterheads provide enough feedback for the nodes, they will be in a position to approximate how much confidence their clusterhead will have in them should they broadcast and hence can decide whether to do so or not. The clusterhead must provide feedback listing the nodes that did broadcast in the previous round and the error margins it believes they have. It will also be necessary, though, to provide the aggregated opinion as feedback so that nodes that do not broadcast can calculate their own error margins and compare it to that of the nodes that did broadcast. With this information nodes can calculate the approximate confidence the clusterhead would have in them by running the same algorithm the clusterhead would but including themselves in the list of nodes that sent readings.

Earlier, in section 4.2, we mentioned that the cluster formation message included other information not related directly to self organisation. We propose including a confidence threshold in that message so that the sink node can approximately control the proportion of nodes broadcasting their readings to clusterheads. In this way the sink node can sacrifice some accuracy to retain energy in the network. We propose that the sink node can release that energy by reforming the network with larger clusters and a lower threshold. The energy saved through nodes not transmitting can be released in this way when the more accurate nodes start to run low on energy. Network reformation is necessary in this case because the clusterheads will be close to running out of energy and new clusterheads need to be chosen.

We propose that reorganisation should include increasing the cluster size to retain a reasonable level of accuracy. During the first phase of the network’s life the most accurate nodes were transmitting meaning that they are now running out of energy leaving only the inaccurate ones to provide data. This will reduce the accuracy of the network. But increasing the cluster size compensates for this loss and can keep the level of accuracy approximately the same. Some testing is needed to find the appropriate levels in different
To achieve our aims the sink must be aware of the status of the network. Therefore, nodes running low on energy should transmit a message to their clusterheads informing them of the situation. Once a certain number of nodes send such a message to the clusterhead it will inform its own clusterhead. In this way the sink node will be informed when the network as a whole is running low on energy. When this happens the sink node can increase the longevity of the network by reforming the network with a larger cluster size and a lower threshold. Our proposal can be seen as creating two or more networks within the sensor network and dynamically moving from one to the other in order to keep the network alive.

The length of time that the network can be kept alive depends on the accuracy that is required. If a low level of accuracy is required then the confidence threshold must be kept low which reduces the number of nodes not transmitting and thus reduces the life-span of the network. As less accuracy is needed the confidence threshold can be raised which would save more energy and increase longevity.
5 Simulation Platform and Procedure

Our simulation was run using the Presage (Programming Environment for the Simulation of Agent Societies) platform [23]. This is a Java based environment that allows the simulation of all parts of the environment including the world and the network. Presage allows for the rapid development of agent societies by providing abstract classes for the world, network and participants that ensure compatibility and also perform the common tasks such as message handling, XML and CSV input files make Presage easily configurable.

Figure 6 shows the constituent parts of Presage.

![Diagram showing the platform architecture of Presage. [23]](image)

Presage simulations are time driven which is appropriate for our scenario and in particular Presage works in discrete time steps which accurately models our timeout based aggregation as presented in section 4.2. Presage also allows agents to schedule actions to be performed at some later time which accords well to our scheduling of aggregating according to level.
Because all components of the agent society being simulated by Presage are extensions of common abstract classes the simulation cycle for Presage is consistent. The cycle first gives the Physical World a chance to do something, in our case produce a value and check the status of the network. Next some behind the scenes clean up is performed removing old conversations. Agents then get a chance to perform some proactive behaviour. We utilise this time for nodes to take their own readings. Following this the agents react to messages that have been sent to them.

Presage uses XML to define protocols and messages conform to the Agent Communication Language standards. A performative in the message is handled by Presage to trigger the execution of the correct handling function. After handling incoming messages, agents get another chance to do something in the world before any messages that were scheduled to be sent that cycle are sent to their recipients. Sending messages is handled by the network which in our scenario uses the strength to determine which nodes receive a copy of the message. That decisions is based on their locations within the physical world which we simulate as a 2D square grid of cells.

We ran three sets of simulations. The first two were proof of concept on individual parts of the solution and the final set were to show that our solution allows us to extend the lifetime of the network by relaxing the accuracy requirement. Throughout, we used a grid of 25x25 cells and set the world to produce a value of 50 every cycle for simplicity. Error margins for nodes were randomly distributed between 0 and 100 and the average error of the nodes was therefore 50.

The first set concerned the improved accuracy from the opinion formation based aggregation algorithm. We ran the simulations for 250 time steps. In one batch we used a single cluster and varied the number of nodes in the network. In the other batch we kept to 100 nodes and varied the cluster size. As we were testing accuracy we ignored the energy consumption. We placed the sink node in the approximate centre of the network to produce a more balanced aggregation tree in the second batch. It would obviously have no effect in the first batch. Every set of variables was simulated once using simple averaging as the aggregation algorithm and once using our proposed algorithm. The average error over the 250 time steps was calculated.

The second set of experiments were designed to show that energy is conserved in the network if we impose a confidence threshold. We used the same grid size and 100 nodes with a fixed cluster size of 25. We varied the threshold over a number of values and recorded the energy left in the network at every time step. We ignored the accuracy at
this stage. Nodes were provided a starting energy of 500 units.

The final set of experiments were to show that using network reformation and various confidence thresholds we could extend the life-span of the network while remaining within certain upper bound error margins. We used 500 nodes each starting with 500 energy units. We set the cluster size to 30 and configured the network so that nodes would report they had low power when they had less than 10% of their starting power. When 50% of the cluster had reported low power the clusterhead would inform its own clusterhead. When network reformation was taking place the cluster size was doubled and the confidence threshold was dropped to zero. For each starting value of the confidence threshold the simulation was run for 750 time steps.
6 Simulation Results

The graphs in figure 7 show the error of the network’s aggregated opinion comparing simple aggregation with our proposed opinion formation aggregation. The first graph shows results from a simulation in which all nodes were one hop from the sink node and hence all formed one large cluster with the sink as clusterhead. We can see that even with simple aggregation the network is more accurate with more nodes, improving from a 5.94% error with 10 nodes to just 1.2% with 200. We also see that the opinion formation aggregation algorithm that we proposed increases the accuracy further giving 3.37% with 10 nodes and 0.61% with 200. The opinion formation aggregation algorithm was between 40% and 50% more accurate than simple averaging for these results.

The second graph introduces the clustering algorithm and shows that larger cluster sizes are more accurate. This is really a corollary of the results from the first graph as that too showed an improvement in accuracy with a larger cluster. With a cluster size of 3, simple averaging had an error of 13.57% while the opinion formation based algorithm had an error of 7.28%. This dropped to 7.62% and 4.24% for a cluster size of 24. Overall, the opinion formation based algorithm was consistently about 45% more accurate.

We noticed that the improvement in accuracy with different cluster sizes was extremely consistent with a variation of less than ±2.5. The 45% improvement was also the improvement found when simulating a network of 100 nodes in a single cluster. This leads us to consider that the relative improvement of the opinion formation algorithm might depend solely on the number of nodes in the network. More work needs to be done to investigate this possibility.

In the first two set of simulations we ignored the power cost in order to analyse the effect on accuracy from the proposed algorithm. In the next set we ignored the accuracy and focused on power. We used 100 nodes with a single cluster and ran the simulation until the first set of nodes lost power. Varying the confidence threshold we recorded the amount of energy left in the network at every time step.

As figure 8 shows, applying a confidence threshold throughout the network results in energy being conserved within it. If there is no threshold then the network completely runs out of power after 240 time steps. At that point, the network still retains almost 20% of its starting power with a confidence threshold of 0.5. This shows that there is energy conservation happening and that it would be possible to release that energy later.
Figure 7: Bar graphs showing the percentage error in the final aggregated opinion of the network comparing simple aggregation with our proposed opinion formation aggregation. The top graph shows results for a network with a single cluster with various number of nodes. The second shows results for a network of 100 nodes using different cluster sizes.
Figure 8: Graph showing the energy remaining in the network over time for different confidence thresholds.
to increase longevity. The graph shows a big jump in the energy being conserved between a threshold of 0.02 and 0.04 and more investigation is warranted in this area with a view to correlating the threshold with an energy saving value.

In the third set of experiments we compared accuracy and longevity. Using 500 nodes with a cluster size of 30 we ran the simulation recording the aggregated opinion at the sink node. Figure 9 compares the result using confidence thresholds of 0.02, 0.03 and 0.04 with the result of using a simple aggregation. The results show that using simple aggregation, with no energy saving and no ability to reform the network, the life-span of the network is 240 time steps and the error is largely below ±10. Using a confidence threshold and with network reformation we can extend the life-span of the network to up to 688 time steps. The result show that our proposed algorithms allow a sensor network to be programmed to live longer by reducing accuracy or to improve accuracy but die quicker.

After the first set of nodes ran out of power we set the network to reform at that point because the clusterheads themselves were soon to run low on power and need replacing. Accuracy suffers after the reformation because the most accurate nodes have lost power and are no longer reporting. Our earlier results show that increasing cluster size improves accuracy and we set the system so that the cluster size would increase after reformation. This is an area that needs further investigation with a view to gaining an understanding of the required cluster size changes for different error bounds.

We also note that network reformation is triggered later with different thresholds (reformation is indicated by large spikes in error during the short time that the sink is the only source of data). We believe that this might result from the adaptive nature of the algorithm. Nodes base their approximation of the confidence that the clusterhead has in them on the error margin it thinks it has and the error margins the clusterhead reports for nodes that sent messages in the previous round. As the more accurate node lose power they no longer send in readings which means they do not feature in the feedback. The other nodes are therefore comparing their error margins to an increasingly decreasing number of other nodes which will likely lead to them believing that the clusterhead will have more confidence in them. Thus, over time the less accurate nodes will start transmitting as the more accurate ones lose power.

This effect is demonstrated in figure 10 which shows the aggregated opinion of a single cluster of 100 nodes over time. No changes were made to the threshold and nevertheless nodes adapted their confidence and started to transmit their readings after the most
Figure 9: Graphs showing the error in the aggregated reading over time under various conditions.
accurate nodes stopped. This property warrants further investigation.

Figure 10: The adaptive nature of the confidence algorithm results in all nodes transmitting at some point regardless of the threshold
7 Conclusion

We started this project noting that in-network aggregation reduces accuracy and that accuracy and energy can be traded off one for the other. We have proposed, developed and implemented an opinion formation based aggregation algorithm that can be used in conjunction with a confidence threshold to allow for the trading off of accuracy for longevity. Our simulations showed that the new aggregation algorithm was itself more accurate than simple averaging by up to 50%. We also showed that for different thresholds we can gain up to a three-fold increase in the life-span of the network.

8 Future Work

We believe there is a lot of scope for future work extending what we have done in this project aside from the specific points mentioned earlier in section 6. One thing that became clear as we progressed through the project is that there are a number factors that affect the performance of the network and for each there are more variables that affect those factors. For example, cluster size was shown to be an important factor in determining accuracy. But cluster size itself is affected by variables we include in the aggregation tree construction algorithm including the values we use for the initial transmission strength, how large the increments are, what maximum we put on transmission strength and, independently of us, the density of the nodes. A useful extension of our work would be to investigate what effect all these factors have in order to build up a better idea of how to achieve some desired result in different circumstances.

Another area worth investigating is the effect of node movement. We have assumed that all nodes are stationary which was a reasonable assumption in our test scenario. However, some applications of sensor networks involve nodes moving freely in their environment, for example if we were interested in the concentration of some contaminant in a moving water supply. Node movement would move nodes out of range of their cluster-heads and into range of others. The logical response would be to periodically reform the network. But clusterheads will have less time to build up approximations of error margins and the robustness of the algorithm to this restriction would be a matter of interest.

It might also be of interest to investigate the impact of specific periods of changed accuracy requirements. Our project focused on situations in which error bounds were set
in advance and kept constant throughout. It would be worth allowing sinks to change
the threshold for some period during the lifetime and investigating what effect this has.

Finally, we would like to consider the effect of using an impact threshold in place of
a confidence one. In this report we used the measure of confidence as the threshold of
whether a node should broadcast its reading. However, it should be possible to save even
more energy by using a threshold that is the product of the confidence and the reading
itself. This would result in nodes not sending readings that are similar to the last reading
even if the clusterhead has large confidence in them. It would also bring into play some
game theoretics as nodes in a cluster need to decide whether other nodes are going to be
transmitting messages in that round.
Appendix A

Power

Our simulation was based on the Mica2Dot sensor node which uses a Chipcon CC1000 transceiver with variable output power. In order to relate the power drawn to the range we used the Path Loss Model. Although the output power is not the actual power drawn in the transceiver we could not find that information and we believe that it doesn’t matter so long as we’re consistent. The Path Loss Model relates the power used in transmission, power required at reception and distance as follows:

\[ P_r = P_t + K - 10\gamma \log_{10}\frac{d}{d_0} \]

where \( P_r \) is the receiving power in dBm, \( P_t \) is the transmission power, \( K \) is some constant given by \( K = 20\log_{10}\frac{\lambda}{4\pi d_0} \), \( \gamma \) is the path loss exponent, \( d_0 \) is the reference distance and \( \lambda \) is the frequency. In our case we have that \( P_r = -110\, dBm, \, d_0 = 1\, m, \lambda = \frac{3\times10^8}{433\times10^6} = 0.69\, m \) and \( K = -25.2 \). We let \( \gamma = 2.5 \) which is a value appropriate for the interior of a large building. We therefore have:

\[-110 = P_t - 25.2 - 25\log_{10}[d] \]
\[25\log_{10}[d] = P_t + 84.8\]

Power in dBm is converted to milliwatts with the following equation:

\[ mW = 10 \, \text{pow}(\frac{dBm}{10}) \]

The data rate is 2.4kbps so that it takes 0.42 milliseconds to send one bit of data. We calculated the size of the messages and from that derived the length of time it would take to transmit them. Combining that with the power drawn in the transmission gives us the energy cost of the message.
Bibliography


