Proc. of Information Processing in Sensor Networks, Palo Alto, CA, April 2003.

Self-Organized Fault-Tolerant Feature Extraction in a Distributed Wireless Sensor Network

Bhaskar Krishnamachari[1] and Sitharama Iyengar[2]

[1] Department of Electrical Engineering
University of Southern California
Los Angeles, California
bhaskark@ieee.org

[2] Department of Computer Science
Louisiana State University
Baton Rouge, Louisiana
iyengar@bit.csc.lsu.edu

Abstract

We consider a canonical task in wireless sensor networks – the extraction of information about environmental features – and propose a multi-step solution that is fault-tolerant, self-organizing and energy-efficient. We explicitly take into account the possibility of sensor measurement faults and develop a distributed Bayesian algorithm for detecting and correcting such faults. Theoretical analysis and simulation results show that 85-95% of faults can be corrected using this algorithm even when as many as 10% of the nodes are faulty. We present a distributed algorithm which combines shortest-path routing mechanisms with leader-election to permit nodes within each feature region to self-organize into routing clusters. These clusters are used in data aggregation schemes that we propose for feature extraction. We show that the best such aggregation scheme can result in an order-of-magnitude improvement in energy savings.

1 Introduction

Wireless sensor networks are envisioned to consist of thousands of devices, each capable of some limited computation, communication and sensing, operating in an unattended mode. According to a recent National Research Council report, the use of such networks of embedded systems "could well dwarf previous revolutions in the information revolution" [45]. These networks are intended for a broad range of environmental sensing applications from vehicle tracking to habitat monitoring [11, 21, 25, 45].

In general sensor networks can be tasked to answer any number of queries about the environment [34]. We focus on one particular class of queries: determining regions in the environment with a distinguishable, "feature" characteristic. As an example, consider a network of devices that are capable of sensing concentrations of some chemical X; an important query in this situation could be "Which regions in the environment have a chemical concentration greater than λ units?" We will refer to the process of getting answers to this type of query as *feature extraction.*

Feature extraction is useful in and of itself as a useful application of a sensor network. It can be considered a canonical task since it is quite easy to envision any number of applications involving environmental sensing

in which this query is extremely important. While feature extraction can certainly be conducted on a static sensor network, it is worthwhile pointing out that it can also be used as a mechanism for non-uniform sensor deployment. If we know where the interesting features are in the operational environment, it may be possible to move or deploy additional sensors to these feature regions in order to get finer grained information.

Wireless sensor networks are often unattended, autonomous systems with severe energy constraints and low-end individual nodes with limited reliability. In such conditions, self-organizing, energy-efficient, fault-tolerant algorithms are required for network operation. These design themes will guide the solution proposed in this paper to the problem of feature extraction.

We begin with a short introduction to some of the prior work in the area of wireless sensor networks, before proceeding to discuss the feature extraction problem and our solution in greater detail.

1.1 Wireless Sensor Networks

A number of independent efforts have been made in recent years to develop the hardware and software architectures needed for wireless sensing. Of particular note are UC Berkeley's Smart Dust Motes [18], TinyOS [17], and the PicoRadio [22] project; the Wireless Integrated Network Sensors (WINS) project [21] and PC-104 based sensors [11] developed at UCLA; and the μAMPS project at MIT [20]. The challenges and design principles involved in networking these devices are discussed in a number of recent works [1, 4, 19, 21, 45].

Self-configuration and self-organizing mechanisms are needed because of the requirement of unattended operation in uncertain, dynamic environments. Some attention has been given to developing localized, distributed, self-configuration mechanisms in sensor networks [12, 31] and studying conditions under which they are feasible [35].

Sensor networks are characterized by severe energy constraints because the nodes will operate with finite battery resources. The energy concerns can be addressed by engineering design at all layers. Some of the energy concerns are being addressed at the hardware and architecture level [18, 32, 36]. At the physical layer, there is now a significant body of work on minimizing energy costs by adjusting the transmit powers of nodes while achieving global network properties such as connectivity [37, 38, 43, 40, 42, 41, 39]. At the link layer, some of the work has focused on energy-efficient medium access schemes suitable for sensor networks [13, 23, 26, 46]. At the networking layer, meta-naming of data and data-aggregation during routing has been proposed and analyzed as a significant means for energy savings [2, 9, 10, 15, 16]. At the application layer, it has been recognized that energy savings can be obtained by pushing computation within the network in the form of localized and distributed algorithms [4, 33, 34].

One of the main advantages of the distributed computing paradigm is that it adds a new dimension of robustness and reliability to computing. Computations done by clusters of independent processors need not be sensitive to the failure of a small portion of the network. Wireless sensor networks are an example of large scale distributed computing systems where fault-tolerance is important. For large scale sensor networks to be economically feasible, the individual nodes necessarily have to be low-end inexpensive devices. Such devices are likely to exhibit unreliable behavior. Therefore it's important to guarantee that faulty behavior of individual components does not affect the overall system behavior. Some of the early work in the area of distributed sensor networks focuses on reliable routing with arbitrary network topologies [28, 29], characterizing sensor fault modalities [5, 6], tolerating faults while performing sensor integration [30], and tolerating faults while ensuring sensor coverage [27]. A mechanism for detecting crash faults in wireless sensor networks is described in [44]. There has been little prior work in the literature on detecting and correcting faults in sensor measurements in an application-specific context.

Thus, the major challenges in sensor networks are the need for unattended autonomous operation, energy efficiency, and fault-tolerance. We will propose a solution to the problem of feature extraction that address

all these challenges. We now discuss this canonical problem and present the component steps in its solution.

1.2 Feature Recognition and Extraction

Consider a wireless network of sensors placed in an operational environment. We wish to task this network to identify the regions in the network that contain interesting features. For example, if the sensors monitor chemical concentrations, then we want to extract the region of the network in which these concentrations are unusually high. This could be true for other sensing modalities as well (high/low temperature regions, high/low acoustic regions etc.). It is assumed that each sensor knows its own geographical location, either through GPS, or through RF-based beacons [47].

It is helpful to treat the trivial centralized solution to the feature recognition problem first in order to understand the shortcomings of such an approach. We could have all nodes report their individual sensor measurements, along with their geographical location directly to a central monitoring node. The processing to determine the feature regions can then be performed centrally. While conceptually simple, this scheme does not scale well with the size of the network due to the communication bottlenecks and energy expenses associated with such a centralized scheme. Hence, we would like a solution in which the nodes in a feature region organize themselves and perform some local processing to determine the extent of the region. This is the approach we will take.

Under ideal conditions, even with the requirement of a distributed, self-organized approach, the problem is not straightforward to solve. However, if we take into account the possibility of sensor measurement faults, there is an additional layer of complexity. If we have unreliable sensors, can they decide on their own if their sensor measurement truly indicates a high "feature" value, or if it is a faulty measurement? In general, this is a difficult question. What if we know that the sensor measurements in the operation region are spatially correlated (since many environmental phenomena are) while sensor faults are uncorrelated? As we establish in this paper, we can exploit such a problem structure to give us a distributed, localized algorithm to mitigate the effect of errors in sensor measurements.

Figure 1 shows a sample scenario. In this situation, we have a grid of sensors in some operational area. There is a feature region with unusually high chemical concentrations. Some of the sensors shown are faulty, in that they report erroneous readings.

We can decompose the process of extracting features in a sensor network into multiple steps, as follows:

1. Determining feature readings: In general, we can think of the sensors measurements as a real number. Assume we are interested in features that correspond to high readings. Now, the sensors need to know what measurement constitutes a feature. Although some work has been done on systems that learn the normal conditions over time so that they can recognize unusual feature readings [48], we consider this issue beyond the scope of this paper. We will instead make the reasonable assumption that a threshold that enables nodes to determine whether their reading corresponds to a feature has been specified with the query, or otherwise made available to the nodes during deployment.

2. Disambiguating "features" from faulty sensor readings: As we indicated before, a challenging task is to disambiguate features from faults in the sensor readings, since an unusually high reading could potentially correspond to both. Conversely, a faulty node may report a low measurement even though it is in a feature region. We will present in section 2 probabilistic decoding mechanisms that exploit the fact that sensor faults are likely to be stochastically uncorrelated, while features are likely to be spatially correlated. In analyzing these schemes, we will show that the impact of faults can be reduced by as much as 85-95% even for reasonably high fault rates.

3. Feature clustering: Once the sensors have determined that they do indeed belong to the feature region, we would like to have them self-organize into a cluster. In section 3 we propose a clustering algorithm that

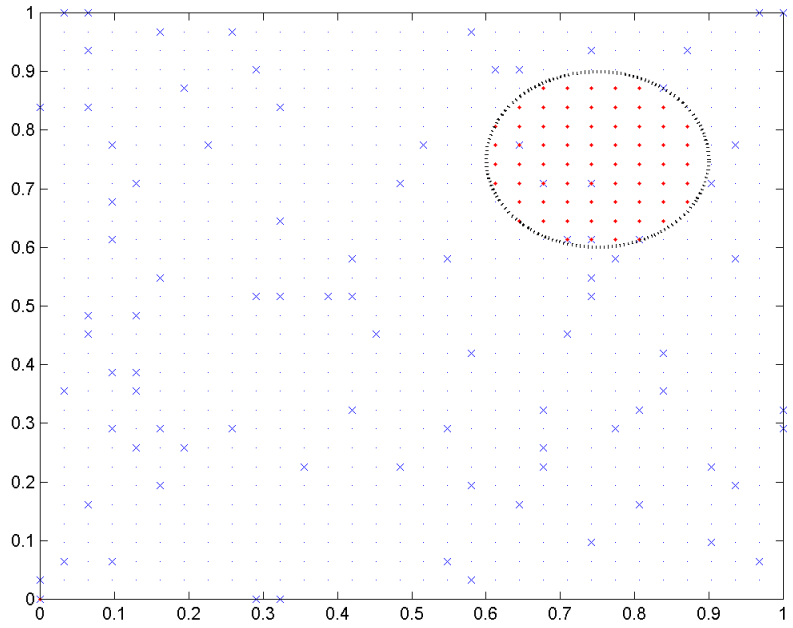

Figure 1: Sample scenario: a distributed sensor network with uncorrelated sensor faults (denoted as 'x') deployed in an environment with a single feature region (dashed circle)

develops intra-cluster routing paths and elects a cluster head that would be responsible for collecting the data for the feature region and routing it to the central data sink.

4. Aggregation/Compression of feature information: Finally, a useful additional step would be to aggregate the data by compressing it in some manner. Sending such a compressed version would save energy resources. We discuss this issue in section 4.

Finally, we will present our conclusions in section 5.

2 Fault-recognition

Without loss of generality, we will assume a model in which a particularly large value is considered unusual, while the normal reading is typically a low value. If we allow for faulty sensors, sometimes such an unusual reading could be the result of a sensor fault, rather than an indication of the feature. We assume environments in which features are typically spread out geographically over multiple contiguous sensors. In such a scenario, we can disambiguate faults from features by examining the correlation in the reading of nearby sensors.

Let the real situation at the sensor node be modelled by a binary variable T_i. This variable $T_i = 0$ if the ground truth is that the node is a normal region, and $T_i = 1$ if the ground truth is that the node is in a "feature" region. We map the real output of the sensor into an abstract binary variable S_i. This variable $S_i = 0$ if the sensor measurement indicates a normal value, and a $S_i = 1$ if it measures an unusual value.

There are thus four possible scenarios: $S_i = 0, T_i = 0$ (sensor correctly reports a normal reading), $S_i = 0, T_i = 1$ (sensor faultily reports a normal reading), $S_i = 1, T_i = 1$ (sensor correctly reports an unusual/feature reading), and $S_i = 1, T_i = 0$ (sensor faultily reports an unusual reading). While each node is aware of the

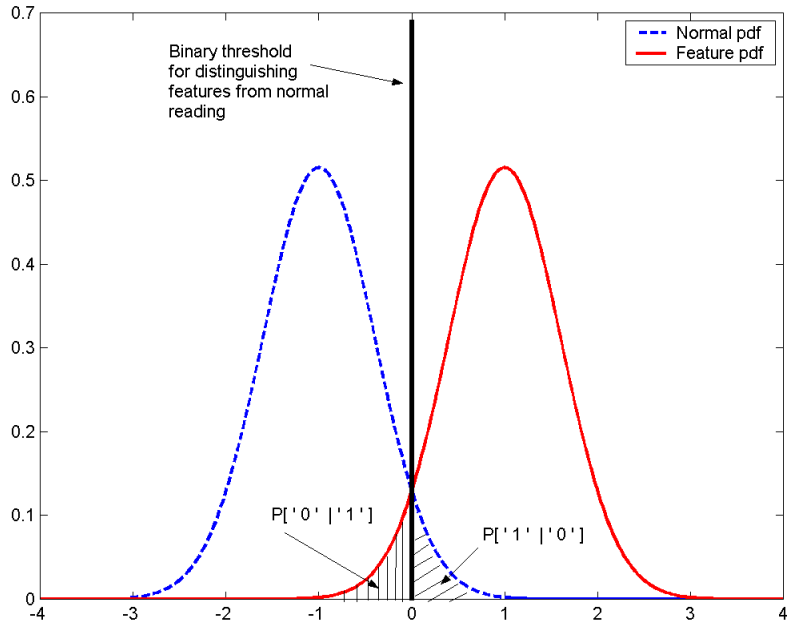

Figure 2: Converting noisy, real-valued, sensor measurements into binary readings

value of S_i, in the presence of a significant probability of a faulty reading, it can happen that $S_i \neq T_i$. We describe below a Bayesian fault-recognition algorithm to determine an estimate R_i of the true reading T_i after obtaining information about the sensor readings of neighboring sensors.

In our discussions, we will make one simplifying assumption: the sensor fault probability p is uncorrelated and symmetric. In other words,

$$P(S_i = 0|T_i = 1) \quad = \quad P(S_i = 1|T_i = 0) = p \tag{1}$$

Figure 2 shows how the binary model can result from placing a threshold on the real-valued readings of sensors. Let m_n be the mean normal reading and m_f the mean feature reading for a sensor. A reasonable threshold for distinguishing between the two possibilities would be $0.5(m_n + m_f)$. If the errors due to sensor faults and the fluctuations in the environment can be modelled by Gaussian distributions with mean 0 and a standard deviation σ, then we get an overlap between the regions as shown in figure 2. In this case the fault probability p would indeed be symmetric and can be evaluated using the tail probability of a Gaussian, the Q-function, as follows:

$$p = Q\left(\frac{(0.5(m_f + m_n) - m_n)}{\sigma}\right) \quad = \quad Q\left(\frac{m_f - m_n}{2\sigma}\right) \tag{2}$$

We know that the Q-function decreases monotonically. Hence (2) tells us that the fault probability is higher when $(m_f - m_n)$ is low (when the mean normal and feature readings are not sufficiently distinguishable) or when the standard deviation σ of the real-valued errors in the sensor readings is high.

We also wish to model the spatial correlation of feature values. Let each node i have N neighbors (excluding itself). Let's say the evidence $E_i(a,k)$ is that k of the neighboring sensors report the same binary reading a as node i , while $N-k$ of them report the reading $\neg a$, then we can decode according to the following rule:

$$P(R_i = a|E_i(a,k)) = \frac{k}{N} \tag{3}$$

Thus we have that each sensor considers that its environmental reality is likely to be that of the average of its neighbors. More sophisticated models are possible, but this model commends itself as a robust mechanism for unforseen environments.

Now, the task for each sensor is to determine a value for R_i given information about its own sensor reading S_i and the evidence $E_i(a,k)$ regarding the readings of its neighbors. The following Bayesian calculations provide the answer:

$$\begin{aligned} P(R_i = a|S_i = b, E_i(a,k)) &= \frac{P(R_i = a, S_i = b|E_i(a,k))}{P(S_i = b|E_i(a,k))} \\ &= \frac{P(S_i = b|R_i = a)P(R_i = a|E_i(a,k))}{P(S_i = b|R_i = a)P(R_i = a|E_i(a,k)) + P(S_i = b|R_i = \neg a)P(R_i = \neg a|E_i(a,k))} \\ &\approx \frac{P(S_i = b|T_i = a)P(R_i = a|E_i(a,k))}{P(S_i = b|T_i = a)P(R_i = a|E_i(a,k)) + P(S_i = b|T_i = \neg a)P(R_i = \neg a|E_i(a,k))} \end{aligned} \tag{4}$$

Where the last relation follows from the fact that R_i is meant to be an estimate of T_i. Thus we have for the two cases $(b = a)$, $(b = \neg a)$:

$$\begin{aligned} P_{aak} = P(R_i = a|S_i = a, E_i(a,k)) &= \frac{(1-p)\frac{k}{N}}{(1-p)\frac{k}{N} + p(1-\frac{k}{N})} \\ &= \frac{(1-p)k}{(1-p)k + p(N-k)} \end{aligned} \tag{5}$$

$$\begin{aligned} P(R_i = \neg a|S_i = a, E_i(a,k)) &= 1 - P(R_i = a|S_i = a, E_i(a,k)) \\ &= \frac{p(N-k)}{(1-p)k + p(N-k)} \end{aligned} \tag{6}$$

Equations (5), (6) show the statistic with which the sensor node can now make a decision about whether or not to disregard its own sensor reading S_i in the face of the evidence $E_i(a,k)$ from its neighbors.

Each node could incorporate randomization and announce if its sensor reading is correct with probability P_{aak}. We will refer to this as the *randomized decision scheme*.

An alternative is a *threshold decision scheme*, which uses a threshold $0 < \Theta < 1$ as follows: if $P(R_i = a|S_i = a, E_i(a,k)) > \Theta$, then R_i is set to a, adn the sensor believes that its sensor reading is correct. If the metric is less than the threshold, then node i decides that its sensor reading is faulty and sets R_i to $\neg a$.

The detailed steps of both schemes are depicted in table 1, along with the optimal threshold decision scheme which we will discuss later in the analysis. It should be noted that with either the randomized decision scheme

Randomized Decision Scheme
1. Obtain the sensor readings S_j of all N_i neighbors of node i.
2. Determine k_i, the number of node i's neighbors j with $S_j = S_i$.
3. Calculate $P_{aak} = \frac{(1-p)k_i}{(1-p)k_i + p(N_i - k_i)}$.
4. Generate a random number $u \in (0, 1)$.
5. If $u < P_{aak}$, set $R_i = S_i$ else set $R_i = \neg S_i$.
Threshold Decision Scheme
1. Obtain the sensor readings S_j of all N_i neighbors of node i.
2. Determine k_i, the number of node i's neighbors j with $S_j = S_i$.
3. Calculate $P_{aak} = \frac{(1-p)k_i}{(1-p)k_i + p(N_i - k_i)}$.
4. If $P_{aak} > \Theta$, set $R_i = S_i$, else set $R_i = \neg S_i$.
Optimal Threshold Decision Scheme
1. Obtain the sensor readings S_j of all N_i neighbors of node i.
2. Determine k_i, the number of node i's neighbors j with $S_j = S_i$.
3. If $k_i \geq 0.5N_i$, set $R_i = S_i$, else set $R_i = \neg S_i$.

Table 1: Decision Schemes for Fault Recognition

or the threshold decision scheme, the relations in 5 and 6 permit the node to also indicate its confidence in the assertion that $R_i = a$.

We now proceed with an analysis of these decoding mechanisms for recognizing and correcting faulty sensor measurements.

2.1 Analysis of Fault-recognition algorithm

In order to simplify the analysis of the Bayesian fault-recognition mechanisms, we will make the assumption that for all N neighbors of node i, the ground truth is the same. In other words, if node i is in a feature region, so are all its neighbors; and if i is not in a feature region, neither are any of its neighbors. This assumption is valid everywhere except at nodes which lie on the boundary of a feature region. For sensor networks with high density, this is a reasonable assumption as the number of such boundary nodes will be relatively small. We will first present results for the randomized decision scheme.

Let g_k be the probability that exactly k of node i's N neighbors are not faulty. This probability is the same irrespective of the value of T_i. This can be readily verified:

$$\begin{aligned} g_k &= \binom{N}{k} P(S_i = 0|T_i = 0)^k P(S_i = 1|T_i = 0)^{(N-k)} \\ &= \binom{N}{k} P(S_i = 1|T_i = 1)^k P(S_i = 0|T_i = 0)^{(N-k)} \\ &= \binom{N}{k} (1-p)^k p^{(N-k)} \end{aligned} \tag{7}$$

With binary values possible for the three variables corresponding to the ground truth T_i, the sensor measurement S_i, and the decoded message R_i, there are eight possible combinations. The conditional probabilities corresponding to these combinations are useful metrics in analyzing the performance of this fault-recognition

Symbol	Definition
n	Total number of deployed nodes.
n_f	Number of nodes in the feature region.
n_o	Number of other nodes $= n - n_f$.
N	The number of neighbors of each node
T_i	The binary variable indicating the ground truth at node i.
S_i	The binary variable indicating the sensor reading. Sensor is faulty $\iff S_i = \neg T_i$.
R_i	The binary variable with the decoded value. Decoding is correct $\iff R_i = T_i$
$E_i(a,k)$	The event that k of node i's N neighbors have the same sensor reading a.
P_{aak}	The conditional probablility $P(R_i = a \mid S_i = a, E_i(a,k))$.
p	The (symmetric) fault probability $P(S_i = 1 \mid T_i = 0) = P(S_i = 0 \mid T_i = 1)$.
g_k	The probability that k of node i's N neighbors are not faulty.
Θ	The decision threshold
α	The average number of errors after decoding
β	The average number of errors corrected
γ	The average number of errors uncorrected
δ	The average number of new errors introduced

Table 2: Summary of Notation

algorithm.

Consider first the probability $P(R_i = 0 \mid S_i = 0, T_i = 0)$. This is the probability that the algorithm estimates that there is no feature reading when the sensor is not faulty and indicates that there is no feature.

$$P(R_i = 0 \mid S_i = 0, T_i = 0) \quad = \quad \sum_{k=0}^{N} P(R_i = 0 \mid S_i = 0, T_i = 0, E_i(0,k)) = \sum_{k=0}^{N} P_{aak} g_k \tag{8}$$

In a similar manner, we can derive the following expressions for all these conditional probabilities:

$$P(R_i = a \mid S_i = a, T_i = a) \quad = \quad 1 - P(R_i = \neg a \mid S_i = a, T_i = a) = \sum_{k=0}^{N} P_{aak} g_k \tag{9}$$

$$P(R_i = \neg a \mid S_i = \neg a, T_i = a) \quad = \quad 1 - P(R_i = a \mid S_i = \neg a, T_i = a) = \sum_{k=0}^{N} P_{aak} g_{N-k} \tag{10}$$

These metrics suffice to answer questions such as the expected number of decoding errors α, obtained by marginalizing over values for S_i.

$$\begin{aligned} \alpha &= P(R_i = 1 \mid T_i = 0) n_o + P(R_i = 0 \mid T_i = 0) n_f \\ &= (1 - \sum_{k=0}^{N} P_{aak}(g_k - g_{N-k})) n \end{aligned} \tag{11}$$

The reduction in the average number of errors is therefore $(np - \alpha)/np$.

We can also now talk meaningfully about β, the average number of sensor faults corrected by the Bayesian fault-recognition algorithm. The conditional probabilities in equations (9) and (10) tell us about this metric:

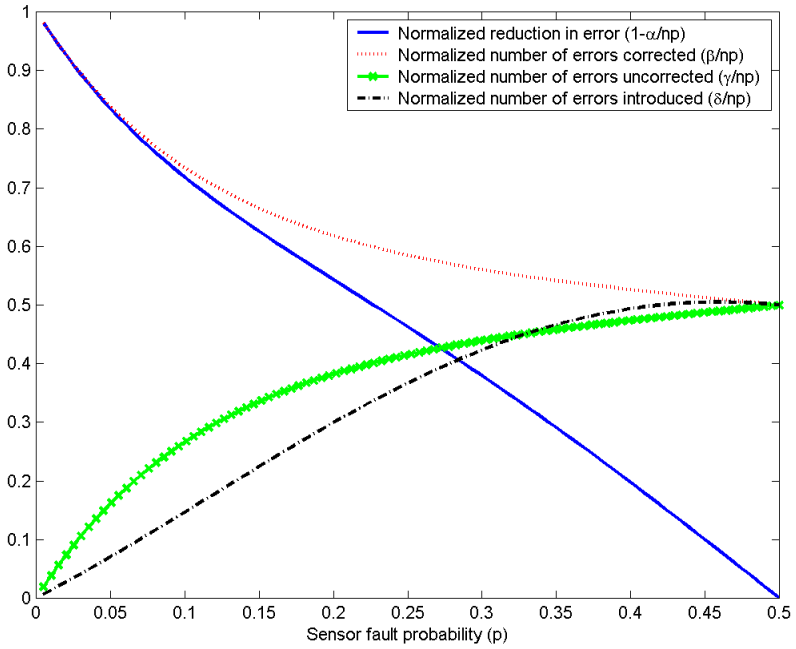

Figure 3: Metrics for the Bayesian fault-recognition algorithm with randomized decision scheme (N = 4)

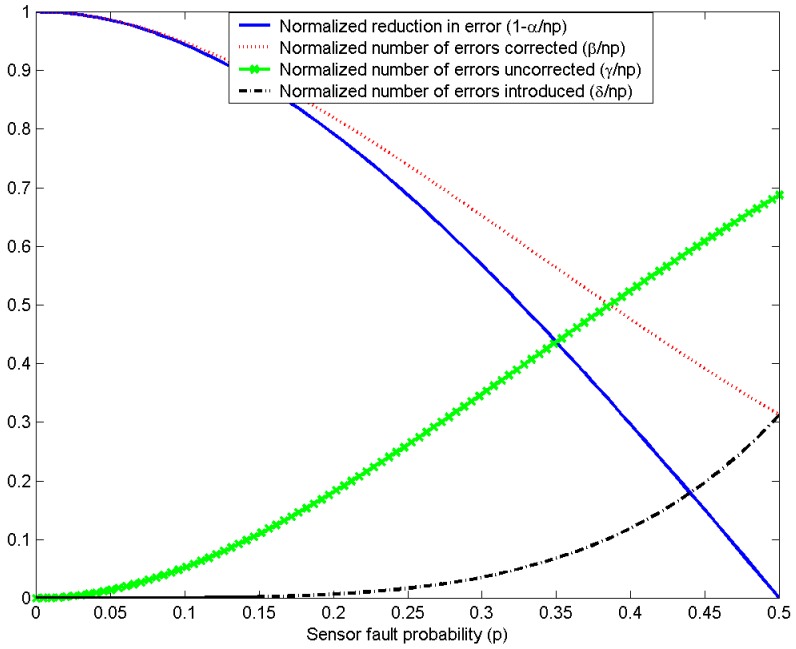

Figure 4: Metrics for the Bayesian fault-recognition algorithm with optimal threshold decision scheme (N=4)

$$\beta \quad = \quad (1 - \sum_{k=0}^{N} P_{aak} g_{N-k}) np \tag{12}$$

A related metric is γ, the average number of faults uncorrected:

$$\gamma \quad = \quad (\sum_{k=0}^{N} P_{aak} g_{N-k}) np \tag{13}$$

The Bayesian fault-recognition algorithm has one setback - while it can help us correct sensor faults, it may introduce new errors if the evidence from neighboring sensors is faulty. This effect can be captured by the metric δ, the average number of new errors introduced by the algorithm:

$$\begin{aligned} \delta &= P(R_i = 1 | S_i = 0, T_i = 0)(1-p) n_o + P(R_i = 0 | S_i = 1, T_i = 1)(1-p) n_f \\ &= (1 - \sum_{k=0}^{N} P_{aak} g_k)(1-p) n \end{aligned} \tag{14}$$

These metrics are shown in figure 3 with respect to the sensor fault probability p. While it can be seen that for $p < 0.1$ (10% of the nodes being faulty on average), over 75% of the faults can be corrected. However, the number of new errors introduced δ is seen to increase steadily with the fault-rate and starts to affect the overall reduction in errors significantly after about $p = 0.1$.

Let us now consider the threshold decision scheme. The following theorem tells us that we can view the threshold scheme from an alternate perspective.

Theorem 1 *The decision threshold scheme with Θ is equivalent to picking a integer k_{min} such that node i decodes to a value $R_i = S_i = a$ if and only if at least k_{min} of its N neighbors report the same sensor measurement a.*

Proof Recall that in this scheme, $R_i = a \iff P_{aak} > \Theta$. It suffices to show that P_{aak} increases monotonically with k, since in this case, for each Θ, there is some k_{min} beyond which R_i is always set to a. We can rewrite equation (5) as follows:

$$P_{aak} \quad = \quad \frac{(1-p)k}{k(1-2p) + pN} \tag{15}$$

The monotonicity can be shown by taking the derivative of this with respect to a continuous version of the variable k:

$$\Rightarrow \frac{d(P_{aak})}{dk} \quad = \quad \frac{p(1-p)N}{(k(1-2p) + pN)^2} > 0 \tag{16}$$

Specifically, k_{min} is given by the following expression, derived by relating equation (15) to the parameter Θ:

$$k_{min} = \left\lceil \frac{pN\Theta}{1-p-(1-2p)\Theta} \right\rceil \tag{17}$$

□

The first question this previous theorem allows us to answer is how the metrics described in equations (8)-(14) change for the decision threshold scheme. In this scheme, we have that if $k \geq k_{min}$ of its neighbors also read the same value a, the node i decides on $R_i = a$. Thus, we can replace P_{aak} in equations (8)-(14) with a step function U_k, which is 1 for $k \geq k_{min}$ and 0 otherwise. This is equivalent to eliminating the P_{aak} term and summing only terms with $k \geq k_{min}$. Thus for the decision threshold scheme we have that:

$$P(R_i = a|S_i = a, T_i = a) = 1 - P(R_i = \neg a|S_i = a, T_i = a) = \sum_{k=k_{min}}^{N} g_k \tag{18}$$

$$P(R_i = \neg a|S_i = \neg a, T_i = a) = 1 - P(R_i = a|S_i = \neg a, T_i = a) = \sum_{k=k_{min}}^{N} g_{N-k} \tag{19}$$

$$\alpha = (1 - \sum_{k=k_{min}}^{N} (g_k - g_{N-k}))n \tag{20}$$

$$\beta = (1 - \sum_{k=k_{min}}^{N} g_{N-k})np \tag{21}$$

$$\gamma = (\sum_{k=k_{min}}^{N} g_{N-k})np \tag{22}$$

$$\delta = (1 - \sum_{k=k_{min}}^{N} g_k)(1-p)n \tag{23}$$

The following is a strong result about the optimal threshold decision scheme.

Theorem 2 *The optimum threshold value which minimizes α, the average number of errors after decoding, is $\Theta^* = (1-p)$. This threshold value corresponds to $k^*_{min} = 0.5N$.*

Proof As the goal is to find the k_{min} and Θ which minimize α, it is helpful to start with the definition of α. From equation (23), we have that:

$$\begin{aligned} \alpha &= (1 - \sum_{k=k_{min}}^{N} (g_k - g_{N-k}))n \\ &= (1 - \sum_{k=k_{min}}^{N} \binom{N}{k}((1-p)^k p^{(N-k)} - p^k(1-p)^{(N-k)}))n \end{aligned} \tag{24}$$

We examine the behavior of the expression in the summand:

$$((1-p)^k p^{(N-k)} - p^k(1-p)^{(N-k)}) \quad = \quad p^k(1-p)^k(p^{(N-2k)} - (1-p)^{(N-2k)}) \tag{25}$$

For $p < 0.5$, this expression is negative for $N > 2k$, zero for $N = 2k$, and positive for $N < 2k$. In the expression for α, as we vary k_{min} by decreasing it by one at a time from N, we get additional terms with negative contributions while $k_{min} > 0.5N$, and positive contributions once $k_{min} < 0.5N$. It follows that α achieves a minimum when $k_{min} = k^*_{min} = 0.5N$.

To determine what value of Θ this corresponds to, we can utilize equation (17). We have that

$$\begin{aligned} \frac{pN\Theta^*}{1-p-(1-2p)\Theta^*} &= 0.5N \\ \Rightarrow p\Theta^* &= 0.5(1-p-(1-2p)\Theta^*) \\ \Rightarrow \Theta^*(p-p+0.5) &= 0.5(1-p) \\ \Rightarrow \Theta^* &= (1-p) \end{aligned} \tag{26}$$

□

The above theorem says that the best policy for each node (in terms of minimizing α, the average number of errors after decoding) is to accept its own sensor reading if and only if at least half of its neighbors have the same reading. This means that the sensor nodes can perform an optimal decision without even having to estimate the value of p. This makes the optimal-threshold decision scheme presented in table 1 an extremely feasible mechanism for minimizing the effect of uncorrelated sensor faults.

2.2 Simulation Results

We conducted some experiments to test the performance of the fault-recognition algorithms. The scenario consists of $n = 1024$ nodes placed in a 32×32 square grid of unit area. The communication radius R determines which neighbors each node can communicate with. R is set to $\frac{1}{\sqrt{n}-1}$, so that each node can only communicate with its immediate neighbor in each cardinal direction. All sensors are binary: they report a '0' to indicate no feature and a '1' to indicate that there is a feature. The faults are modelled by the uncorrelated, symmetric, Bernoulli random variable. Thus each node has an independent probability p of reporting a '0' as a '1' or vice versa. We model correlated features by having l single point-sources placed in the area, and assuming that all nodes within radius S of each point-source have a ground truth reading of 1, i.e. detect a feature if they are not faulty. For the scenario for which the simulation results are presented here, $l = 1, S = 0.15$.

We now describe the simulation results. The most significant way in which the simulations differ from the theoretical analysis that we have presented thus far is that the theoretical analysis ignored edge and boundary effects. This can play a role because at the edge of the deployed network, the number of neighbors per node is less than that in the interior, and also the nodes at the edge of a feature region are more likely to erroneously determine their reading if their neighbors provide conflicting information. Such boundary nodes are the most likely sites of new errors introduced by the fault-recognition algorithms presented above. In general, because of this, we would expect the number of newly introduced errors to be higher than that predicted by the analysis.

Figure 5 shows a snapshot of the results of a sample simulation run. The sensor nodes are depicted by dots; the nodes indicated with bold dots are part of the circular feature region. An 'x' indicates a faulty node (before the fault-recognition algorithm), while an 'o' indicates a node with erroneous readings after

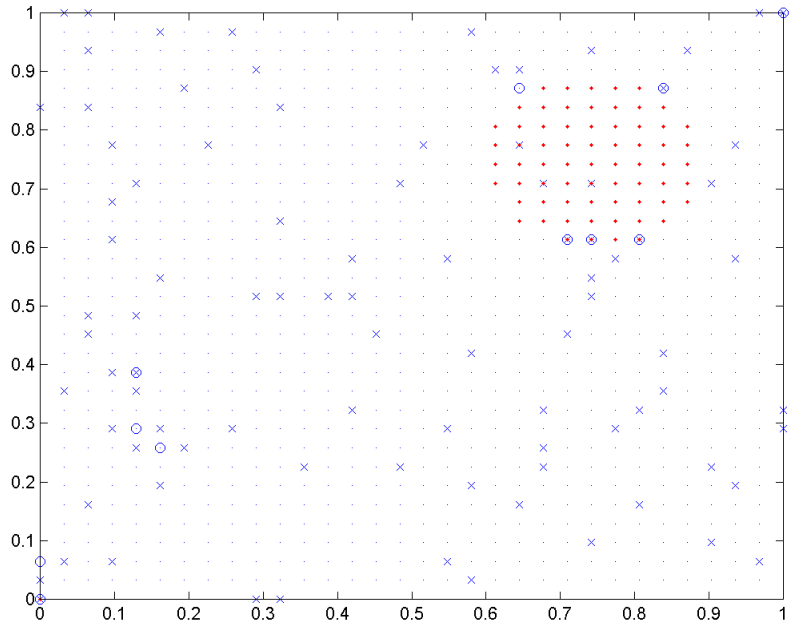

Figure 5: A snapshot of the simulator showing the errors before and after fault-recognition with optimal threshold (p = 0.1)

fault-recognition. Thus nodes with both an 'x' and 'o' are nodes whose errors were not corrected, while node with an 'x' but no 'o' are nodes whose errors were corrected, and nodes with no 'x', but an 'o' are nodes where new error has been introduced by the fault recognition algorithm. Most of the remaining errors are concentrated on the boundaries of the feature region.

Figures 6, 7, and 8, show the important performance measures for the fault recognition algorithm with the optimal threshold decision scheme from both the simulation as well as the theoretical equations. The key conclusion from these plots is that the simulation matches the theoretical predictions closely in all respects except the statistic of newly introduced errors, where understandably the border effects in the simulation result in higher values. More concretely, these figures show that well over 85% - 95% of the faults can be corrected even when the fault rate is as high as 10% of the entire network.

Figure 9 illustrates the performance of the threshold decision scheme with respect to the threshold value Θ. Again, the simulation and theoretical predictions are in close agreement. The optimal value of the threshold Θ is indeed found to correspond to a k_{min} of $0.5N$.

3 Self-organized Feature Clustering

Once the feature nodes have been identified by the fault-recognition algorithm, we would like to have these nodes self-organize into clusters and elect cluster heads to enable local information processing. We propose to achieve this by combining a distributed election leader algorithm [7] with a distance-vector routing [8] mechanism. The combination is an algorithm in which the immediate neighbors of the leader get the correct information first, then the neighbors of these neighbors, and so on until all nodes within the cluster obtain a path to the same cluster leader. We now give details of this clustering algorithm.

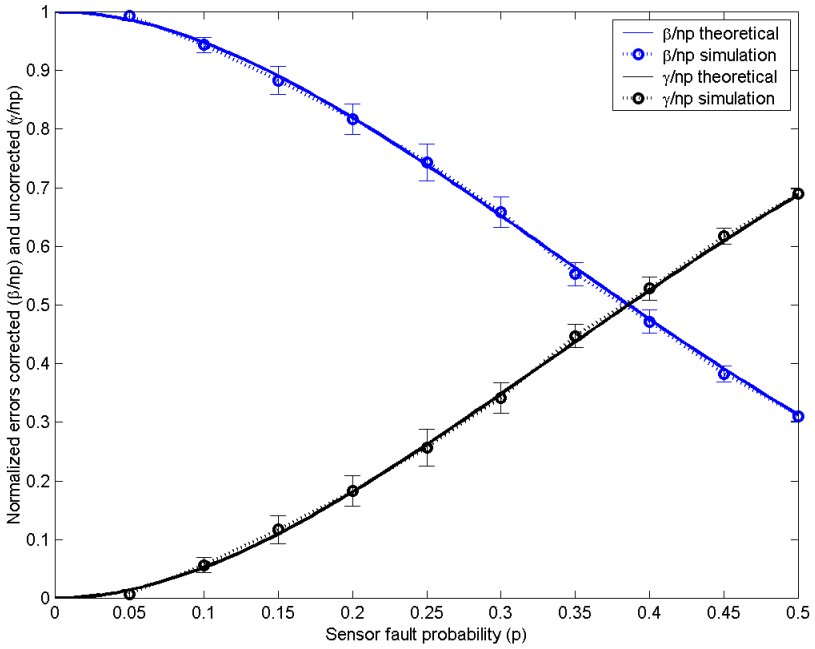

Figure 6: Normalized number of errors corrected and uncorrected with the optimal threshold decision scheme

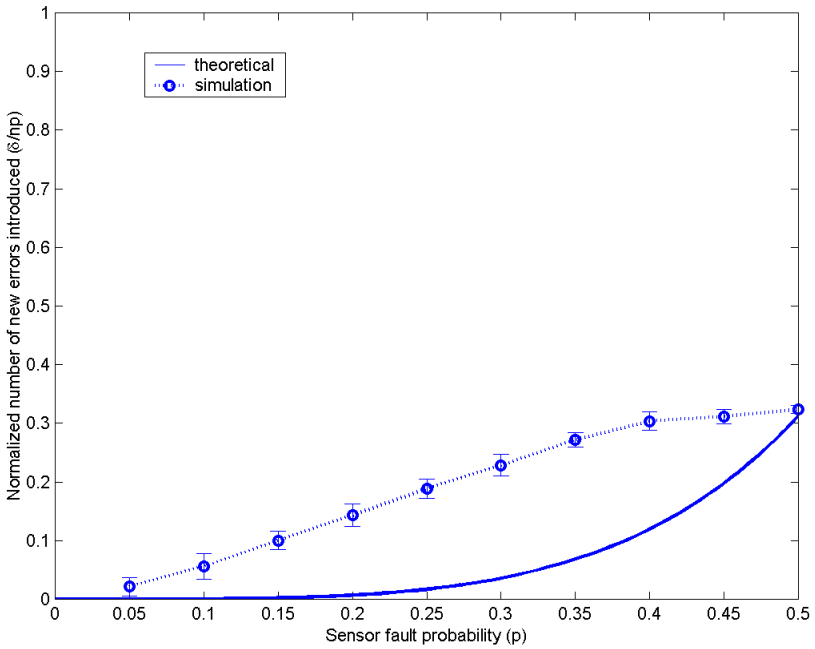

Figure 7: Normalized number of new errors introduced with the optimal threshold decision scheme

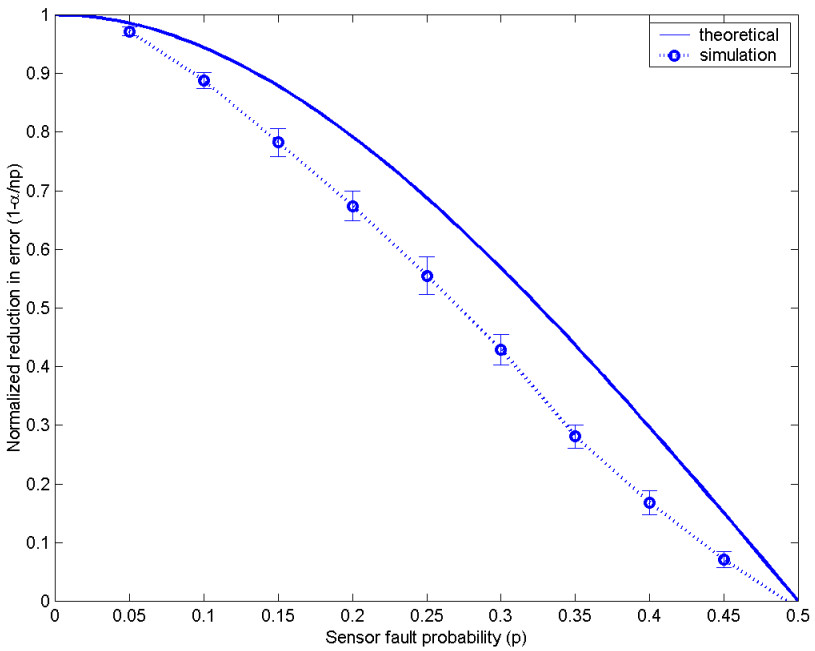

Figure 8: Normalized reduction in average number of errors for the optimal threshold decision scheme

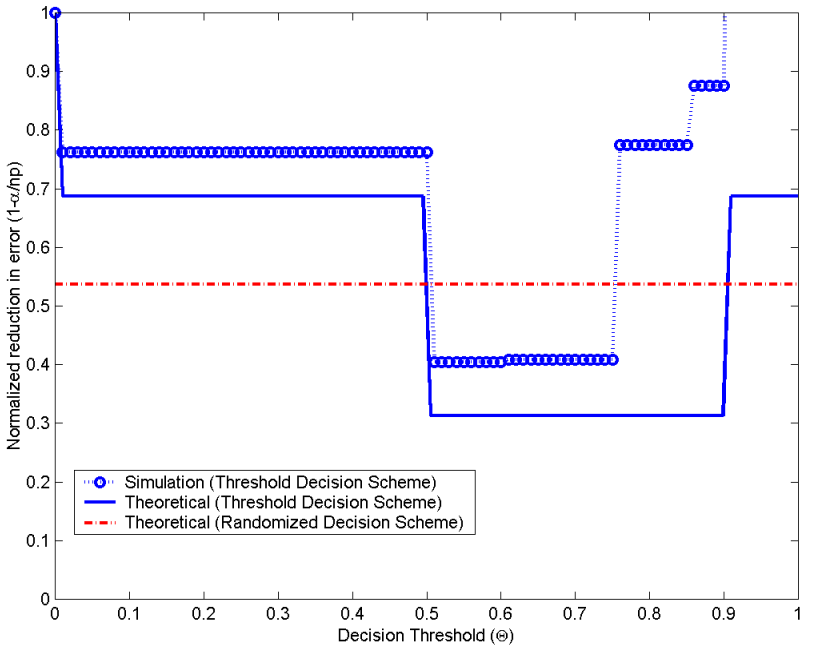

Figure 9: Normalized reduction in average number of errors with respect to the threshold value in the threshold decision scheme (p = 0.25, $\Theta* = 1 - p = 0.75$)

Only nodes which have a feature reading participate in this mechanism. As with most leader election algorithms, it is assumed that each node i within the cluster has a unique ID value ID_i that can be used to determine the cluster head (typically the lowest ID number node is elected, though this can be modified for some other metric easily). One useful way in which unique ID's can be chosen is to base their value on the geographical location of the nodes, particularly on their distance to the central monitoring node. This is likely to result in the cluster head being close to the data sink.

Each node i maintains a 3-tuple (LID_i, K_i, NH_i). The field LID_i is the lowest ID number seen to date by node i; K_i is the number of hops from i to the node with lowest ID; and NH_i is the next hop from i towards the lowest ID node. Initially, for all i, $LID_i = ID_i, K_i = 0$ and the NH_i field is left blank.

3.1 Description of feature clustering algorithm:

1. Do the following if and only if $R_i = 1$, i.e. if i is a feature node.
2. `if` i `has an previously uncommunicated value for (`LID_i`,` K_i`,` NH_i`), communicate this to all neighbors`
3. i `examines any tuples received from neighbors` j `and updates information as follows:`

```
if LID_j < LID_i then
     LID_i = LID_j
     K_i = K_j+1
     NH_i = j
  else if LID_j == LID_i then
      if K_j < K_i − 1
         K_i = K_j + 1
         NH_i = j
      endif
  endif
```

3.2 Analysis of the feature clustering algorithm

Correctness and complexity of algorithm:

We first consider the synchronous version of the feature clustering algorithm (an assumption that can be relaxed). The algorithm proceeds in multiple rounds. Each node sends at most one message at each round. The following theorem tells us about the correctness and complexity:

Theorem 3 *If D is the diameter of the cluster (i.e. the maximum hop distance between any pair of nodes in the feature region) the algorithm is guaranteed to converge in at most D steps, and each node issues no more than D messages. At the termination of the feature clustering algorithm, for all i, $LID_i = \min_j(ID_j)$, and the entries K_i and NH_i are consistent with the shortest path between node i and the elected leader node l which has $ID_l = LID_i$.*

Proof: Given synchronous operation, it is easiest to prove this by induction.

Base case: Consider the node l which has the lowest ID, i.e. the node which should be elected the leader at the end of the process. It is easy to see that at the end of the first round, all nodes i within one hop of S

update their table so that $LID_i = ID_l$, $K_i = 1$, and $NH_i = l$. From this point on, these entries are frozen, i.e. will never change.

Induction hypothesis: Assume that at the end of the t^{th} round all nodes i that are exactly t hops away from the node l have just updated their table so that $LID_i = ID_l$, $K_i = t$, and $NH_i = j$, where $K_j = t - 1$. It is clear that these entries are frozen, as per the description of the algorithm.

Induction step: Any node i that is exactly $t + 1$ hops away from l must neighbor some nodes J which are t hops away from l. At the end of round $(t + 1)$, i hears from nodes in J since they have had a change in their entry at time step t. From the first $j \in J$ that i hears from, it updates its values so that $LID_i = ID_l$, $K_i = t + 1$, $NH_i = j$. This entry is henceforth frozen.

Thus, since all nodes are at most distance D from the source l, all nodes will have their entries frozen in at most D steps, at which point the algorithm terminates. Further, since each node only issues at most one message in each round, no node issues more than D messages.

□

Synchronization Requirements: If we have perfect synchronization of the clocks of all nodes, then they all can have a common idea of when each round is supposed to begin and end. The algorithm can also be carried out with only a limited degree of synchrony if we require that all nodes transmit at each step.

In order to relax the synchronization assumption, we require each node in the feature region to send the initial value of its entry to all its neighbors within a finite time T of the start of the clustering algorithm (which can be defined as the time when the feature extraction query is received by all nodes in the cluster). Thereafter, we require each node to wait until it has heard from all its neighbors before calculating and sending its updated values to all its neighbors and proceeding to the next round, and to take no more than time T_2 between the time it has heard from all its neighbors and the time it has calculated and if necessary, sent the updated values to all its neighbors. In this case, all nodes send the first step values to all their neighbors by time T, the second step values by time $T + T_2$, and so on. Thus even if they are not all synchronized to the same clock value, they can synchronize their behavior to the level of a step by relying their internal clocks alone. Since we now require all nodes to send information at each step, they must all be aware of a bound on D and the value of T and T_2 so that they can safely terminate the algorithm at time $T + (D - 1)T_2$.

3.3 Illustration

This proposed clustering mechanism is best explained by an illustration. In figure 10, we see a single run of this algorithm in a feature region containing five nodes labelled with relevant IDs. Note that all nearby sensor nodes that do not have feature readings (shown in dashed lines) do not participate. As the algorithm progresses, the shaded nodes have obtained the correct information and do not change values thereafter. The progress of these shaded nodes shows the clustering algorithm's characteristic of having the correct information spreading outward from a single node level by level.

At the conclusion of this cluster formation mechanism we have a spanning tree whereby each node can pass information on to the cluster head. Figure 11 shows a snapshot from the simulator depicting the intra-cluster spanning tree and election of cluster head in the feature region for our sample scenario.

Note the distributed, self-organizing, nature of the entire process - no central commands need to be issued to determine the cluster head for each region and to perform the intra-cluster routing setup. The entire process can be triggered automatically when the feature readings are determined. The algorithm is highly robust to the addition of new nodes to the feature region: any new node i initially advertises its tuple to be (i, 0, -), and the neighbors of this node which are in the feature region would respond by with their

current values. If the new node is not going to be the new cluster leader (because its ID number is not low enough), then no additional messages need to be exchanged. If the new node should be the cluster leader, then the clustering algorithm starts afresh in the entire region. It is assumed that complete node failures are rare in the network, but it should be noted that some form of refresh mechanism is required to ensure that the intra-cluster routing information does not become stale. Another approach could be the use of link reversal mechanisms to deal with such failures in the presence of extreme dynamics [49]. Finally, we note that the clustering algorithm can be conducted in parallel throughout the network, resulting in the formation of multiple independent clusters simultaneously in separate feature regions.

4 Data Aggregation Mechanisms

As we mentioned before, energy conservation is necessary to extend the lifetime of wireless sensor networks because they are unattended and may have limited battery resources. Cross-layer optimizations can provide the needed savings.

One cross-layer methodology that has found to be particularly useful is in-network processing, which combines application and networking layers. The intermediate nodes, which act as routers in the multi-hop network, are also permitted to read the packets and process their application-level contents. Such techniques were first developed to allow for greater networking flexibility in Active Networks [53] and router-assist for internet multicast [52]. In sensor networks, the form of in-network processing which is useful for energy conservation is data aggregation. The basic idea behind data aggregation is simple: data coming from multiple sources enroute to the same destination can be combined. This eliminates redundant communication within the network, a significant source of the energy expenditure.

Directed Diffusion is an example of a data-centric routing protocol for sensor networks that incorporates data-aggregation explicitly in its functioning [2]. Experimental and analytical results on data-centric routing suggest that significant energy savings can be obtained using aggregation techniques [14, 9, 3]. [10] discusses the implementation of aggregation for optimizing responses to database-type queries in sensor networks. Most previous work has dealt with exact aggregation mechanisms such as duplicate suppression, min, max, sum, and count. We will discuss here some data-aggregation techniques appropriate to the feature extraction problem, including some approximate aggregation schemes.

The following are some pieces of information the central monitoring node may query for:

- the location of the feature region(s)
- the readings of each individual node in the feature region
- min/max/average reading in the feature region

Different forms of aggregation may be suitable depending on the query. We focus on the location query. This query says "describe the location of the feature region." The first thing to note is that this is a query that lends itself to both exact and inexact answers. On the one hand we can report the full detailed information of the locations of the nodes in the feature region; on the other hand we can give an approximate parametric description that loosely describes a geometric shape containing all nodes in the feature region.

Let us assume that each node knows the x and y-coordinates for its own location. We also assume that all packets have a fixed header size H, and use B bits to represent each coordinate of a location. Let k refer to the total number of sources, i.e. nodes in the feature region, d the distance in hops from the cluster-head to the sink, and d_i, the shortest distance between the cluster-head and the i^{th} node in the feature region. The following are some aggregation options that can be pursued:

No Aggregation (NA): If no aggregation mechanism is employed, then each of the k node will send 2B bits of location information (to indicate its x-coordinate and y-coordinate) plus a header of size H along a total $d + d_i$ hops. Thus the energy cost of this scheme in terms of the total number of bits transmitted in this case λ_{NA} will be

$$\lambda_{NA} = (2B + H)(kd + \sum d_i) \tag{27}$$

Header Aggregation (HA): In this scheme, all nodes in the feature region send their location information in separate packets through the intra-cluster routing tree to the cluster-head. The cluster-head then combines all these packets without any modification into one large packet. This scheme can provide modest savings in cost, particularly if the header size in each packet is comparable in size to the data. The total number of bits transmitted in this case, λ_{HA} is given by:

$$\lambda_{HA} = 2B(kd + \sum d_i) + H(d + \sum d_i) \tag{28}$$

Header Aggregation with Lossless Compression (HAC): An additional level of savings can be obtained in the header aggregation scheme, if the cluster-head compresses the information it obtains from all nodes in the feature region before sending it on. For exact location information about each node to be preserved, a lossless source-coding scheme such as the Lempel-Ziv algorithm may be used. Assuming the original data can be compressed by a factor of $\rho \leq 1$, we have that the total number of bits transmitted in this case is

$$\lambda_{HAC} = 2B(kd\rho + \sum d_i) + H(d + \sum d_i) \tag{29}$$

Rectangular Approximate Aggregation (RA): The previous schemes all focus on sending the exact location details about each node in the feature region. However, if it suffices to know the approximate location and extent of the feature region, significant reduction can be obtained by combining the information into a geometric shape such as the rectangle which contains the nodes in the feature region. The cluster-head collects (x,y) coordinates for all such nodes, and first computes the 4-tuple $[XMIN, YMIN, XMAX, YMAX]$ from the collected values. It then calculates the length of the diagonal $DIAG$, which is simply the Euclidean distance between $(XMIN, YMIN)$ and $(XMAX, YMAX)$.

The cluster-head then sends the 3-tuple $[XMIN, YMIN, DIAG]$ (which suffices to reconstruct the enclosing rectangle) on to the central monitoring node. Since only $3B$ bits of information need to be sent from the cluster-head to the central monitoring node, the number of bits transmitted in this scheme is given as

$$\lambda_{RA} = 2B \sum d_i + 3Bd + H(d + \sum d_i) \tag{30}$$

Circular Approximate Aggregation (CA): This scheme is similar to the rectangular approximate aggregation, except that the cluster-head instead computes the center and radius of the smallest circle which encloses all nodes in the feature region, represented as the 3-tuple $[XMID, YMID, RADIUS]$. The computation of this circle, known as the Minimum Enclosing Circle problem, has a long history dating back to the mid 19th century when the first simple $O(n^3)$ algorithms were proposed. A series of new algorithms in the 70s brought the complexity of this computation down to $O(nlogn)$. A landmark result in computational geometry was the development of ingenious prune-and-search linear programming techniques by Megiddo and Dyer [50, 51] which allows this minimum enclosing circle to be computed in $O(n)$ time. Since only three parameters requiring $3B$ bits need to be sent from the cluster-head, the number of bits transmitted in this scheme is given as

Scheme	Bits Used	Savings	Response Quality
No aggregation (NA)	221544	0%	Exact
Header Aggregation (HA)	117544	46.9 %	Exact
HA with Compression (HAC)	100648	54.6 %	Exact
Rectangular Aggregation (RA)	34984	84.2 %	Tight rect. approximation
Circular Aggregation (CA)	34984	84.2%	Tight circ. approximation
Stepwise Rectangular Aggregation (SRA)	9240	95.8%	Tight rect. approximation
Stepwise Circular Aggregation (SCA)	9240	95.8%	Loose circ. approximation

Table 3: Comparison of various aggregation schemes for sample simulated scenario

$$\lambda_{CA} = 2B \sum d_i + 3Bd + H(d + \sum d_i) \tag{31}$$

Step-wise Rectangular Aggregation (SRA): If we permit each node within the cluster to aggregate the information coming from all downstream nodes, we can get further gains with the rectangular aggregation scheme. The idea here is simple, each node maintains the 4-tuple $[XMIN, XMAX, YMIN, YMAX]$ and the correspondingly computed 3-tuple $[XMIN, XMAX, DIAG]$. To begin, leaf nodes send this information upstream in the form of the 3-tuple. From this point on, nodes wait to collect this information from all their downstream nodes, and compute the value of the rectangle that contains themselves and all their downstream nodes. They then send the corresponding 3-tuple upstream. This is illustrated in figure 12. Since exactly $3B$ bits are sent at each hop, we have that the energy cost of this scheme is

$$\lambda_{SRA} = 3B(k + d - 1) + H(d + \sum d_i) \tag{32}$$

It is important to note that the final information obtained by the central monitoring node is the same in the case of RA as well as SRA aggregation schemes. In both cases, it obtains the coordinates of the smallest rectangle enclosing the feature nodes.

Step-wise Circular Aggregation (SCA): This scheme is similar to the SRA. Each node sends the 3-tuple $[XMID, YMID, RADIUS]$ upstream. This tuple is used to describe the center and radius of the smallest circular region that includes the intersection of the circular regions of its descendant nodes as well as its own location. As this information makes it way upstream, the final result which the cluster head obtains are the coordinates for the center as well as the radius of a circular region which contains all the nodes in the feature region. However, it is important to note that the approximation can become less and less accurate as the information makes its way upstream. Specifically the circular region that the cluster-head obtains with this scheme is in general an overestimate of the minimum enclosing circle. As a 3-tuple is used at each hop, the total number of bits expended by this method is again:

$$\lambda_{SCA} = 3B(k + d - 1) + H(d + \sum d_i) \tag{33}$$

Table 3 shows a comparison of the above schemes on the sample simulation scenario of figures 1, 5, and 11. In this simulation the values for the size parameters were $H = 40$ and $B = 16$. For HAC, the compression ratio was set to a typical value of $\rho = 0.8$. The number of nodes in the cluster is $k = 66$, the distance between the cluster-head and the central monitoring node is $d = 40$, and the sum of the intra-cluster distances was evaluated to be $\sum d_i = 437$. We can see that since the header size is comparable to the size of the data contents, even header aggregation can reduce the energy costs by nearly half in this case. As noted before, the first three schemes all provide exact information about the location of each individual node, while the remaining schemes provide some form of approximation. Both the RA and CA schemes result in nearly 85% energy savings in this scenario, the additional gains coming chiefly due to the reduction of data being sent from from the cluster-head to the central monitoring node. Both approximations are tight, in the sense that

they provide the coordinates of the minimum enclosing rectangle and circle respectively. For applications where the feature is likely to be approximately circular in shape (for example if the chemical concentrations in the environment diffuse uniformly in all directions), the circular approximation may be closer to the real situation. However, when we consider the two step-wise approximate aggregation schemes, the SRA scheme is better since it still provides the minimal enclosing rectangle with significant savings (95% in this scenario), whereas the SCA scheme (which incurs the same costs) can result in a loose overestimate of the region containing all the nodes. Overall, we can conclude the SRA scheme is a robust, reasonably tight approximate aggregation algorithm which provides the most energy gains for this application.

One additional fact that should be pointed out about the approximate aggregation schemes is that because they provide information about enclosing regions, they are somewhat robust to faults in the boundary of the feature region. Since most of the faults that remain after the fault-correction algorithms we described in section 2 are likely to be on the boundaries, this is a beneficial property of the approximate aggregation schemes.

It's also insightful to look at the limiting behavior of the energy gains. An upper-bound on the energy gains is obtained if we let d tend to infinity (when the feature region is really far away from the central monitoring node).

Theorem 4 *If $max(d_i)$ and k are fixed then*

$$\lim_{d\to\infty} 1 - \frac{\lambda_{SRA}}{\lambda_{NA}} = 1 - \frac{(3B+H)}{(2B+H)k} \tag{34}$$

Proof: From equations (27) and (32), we get that

$$\frac{\lambda_{SRA}}{\lambda_{NA}} = \frac{3B(k+d-1) + H(d+\sum d_i)}{(2B+H)(kd+\sum d_i)} \tag{35}$$

$$\Rightarrow \lim_{d\to\infty} \frac{\lambda_{SRA}}{\lambda_{NA}} = \frac{(3B+H)d + 3B(k-1) + H\sum d_i}{(2B+H)kd + (2B+H)\sum d_i} \tag{36}$$

$$\Rightarrow \lim_{d\to\infty} \frac{\lambda_{SRA}}{\lambda_{NA}} = \frac{(3B+H)d + o(1)}{(2B+H)kd + o(1)} \tag{37}$$

$$\Rightarrow \lim_{d\to\infty} 1 - \frac{\lambda_{SRA}}{\lambda_{NA}} = 1 - \frac{(3B+H)}{(2B+H)k} \tag{38}$$

□

Theorem 4 represents an upper bound on the gains that can be obtained with SRA. For the sample scenario that we studied, if we let $d \to \infty$, we get a gain of $(1 - 88/(72 \cdot 66))100 = 98.2\%$, which is close to the 95.8% achieved in the simulation for $d = 40$.

The above schemes can be generalized for other queries sent to the feature nodes. For detailed and exact information such as IDs or readings of all nodes in the feature region, approximate schemes are invalid. Thus the HA and HAC schemes are the most appropiate. For queries which require a single number to be sent back, such as query asking for the min/max/average feature reading, a suitable approach is to perform in-cluster aggregation. The information could either be aggregated at the cluster-head after receiving all inputs directly from each node in the cluster, or in a stepwise manner throughout the cluster. The cluster-head then sends this single number on to the central monitoring node. The energy gains through these schemes would be quite comparable to those obtained for the feature-location query with the RA and SRA schemes, respectively.

5 Conclusions

With recent advances in technology it has become feasible to consider the deployment of large-scale wireless sensor networks that can provide high-quality environmental monitoring for a range of applications. In this paper we developed a multi-stage solution to a canonical task in such networks – the extraction of information about regions in the environment with identifiable features.

In such networks involving thousands of unattended, low-cost, low-capability devices, reliability, self-organization, and energy-efficiency are paramount concerns. Our solution effectively addresses all these concerns, and illustrates the design principles for this emerging space of application-specific networks.

One of the most difficult challenge is that of distinguishing between faulty sensor measurements and unusual environmental conditions. To our knowledge, this is the first paper to propose a solution to the fault-feature disambiguation problem in sensor networks. Our proposed solution, in the form of Bayesian fault-recognition algorithms, exploits the notion that measurement errors due to faulty equipment are likely to be uncorrelated, while environmental conditions are spatially correlated.

We presented two Bayesian algorithms, the randomized decision scheme and the threshold decision scheme and derived analytical expressions for their performance. Our analysis showed that the threshold decision scheme has better performance in terms of the minimization of errors. We also derived the optimal setting for the threshold decision scheme for the average-correlation model. The proposed algorithm has the additional advantage of being completely distributed and localized - each node only needs to obtain information from neighboring sensors in order to make its decisions. The theoretical and simulation results show that with the optimal threshold decision scheme, faults can be reduced by as much as 85 to 95% for fault rates as high as 10%.

We then presented a distributed mechanism for nodes in a feature region to self-organize into a cluster. The proposed mechanism combines shortest-path routing techniques with a leader-election mechanism. The final result of the clustering algorithm is the election of a cluster-head and the formation of a minimum spanning tree connecting all the other nodes to the cluster-head.

This cluster is then used as a precursor for in-network processing when information about the feature region is extracted back to the central monitoring node. We presented and analyzed a number of distinct data-aggregation mechanisms to provide energy savings by the elimination of redundant information. We showed that one of these schemes, the stepwise rectangular approximation scheme has the advantage of being robust to boundary-errors in the fault recognition algorithm, providing a tight approximation of the feature region, and resulting in order-of-magnitude savings in energy costs. For the simulated scenario, for example, this saving was over 95%.

There are a number of directions in which this work can be extended. The most promising is the extension of our work on fault-recognition and fault-tolerance in sensor networks. We have dealt with a binary fault-feature disambiguation problem here. This could be generalized to a notion of the correction of real-valued sensor measurement errors: nodes in a sensor network should be able to exploit the spatial correlation of environmental readings to correct for the noise in their readings. Another related direction is to consider dynamic sensor faults where the same nodes need not always be faulty. Much of the work presented here can also be easily extended to dynamic feature recognition to deal with environmental phenomena that change location / shape over time. We would also like to see the algorithms proposed in this paper implemented and validated on real sensor network hardware in the near future.

References

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks," *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May 2001.

[2] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 2000)*, August 2000, Boston, Massachusetts

[3] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks" , In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria. July, 2002.

[4] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom '99)*, Seattle, Washington, August 1999.

[5] K. Marzullo, "Implementing fault-tolerant sensors," TR89-997, Dept. of Computer Science, Cornell University, may 1989.

[6] L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan, "Functional Characterization of Fault Tolerant Interation in Disributed Sensor Networks," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 21, No. 5, September/October 1991.

[7] N. Lynch, *Distributed Algorithms*

[8] B.A. Forouzan, *Data Communications and Networking*, McGraw Hill, 2001.

[9] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of Data Aggregation in Wireless Sensor Networks," *International Workshop on Distributed Event Based Systems, DEBS'02*, July 2002.

[10] S. Madden, R. Szewczyk, M. Franklin, and D. Culler, "Supporting Aggregate Queries over Ad-Hoc Wireless Sensor Networks," *IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[11] A. Cerpa *et al.*, "Habitat monitoring: Application driver for wireless communications technology," *2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[12] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies," *INFOCOM*, 2002.

[13] Seong-Hwan Cho and A. Chandrakasan, "Energy Efficient Protocols for Low Duty Cycle Wireless Microsensor Networks", *ICASSP 2001*, May 2001.

[14] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming," *18th ACM Symposium on Operating Systems Principles*, October 21-24, 2001.

[15] W.R. Heinzelman, J. Kulik, and H. Balakrishnan "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, Seattle, Washington, August 15-20, 1999, pp. 174-185.

[16] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *33rd International Conference on System Sciences (HICSS '00)*, January 2000.

[17] J. Hill *et al.*, "System Architecture Directions for Networked Sensors," *ASPLOS*, 2000.

[18] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 99)*, Seattle, WA, August 17-19, 1999

[19] W.W. Manges, "Wireless Sensor Network Topologies," *Sensors Magazine*, vol. 17, no. 5, May 2000.

[20] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Amit Sinha, Eugene Shih, Alice Wang, and Anantha Chandrakasan, "Low-Power Wireless Sensor Networks", *VLSI Design 2000*, January 2001.

[21] G.J. Pottie, W.J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 551-8, May 2000.

[22] J. Rabaey, J. Ammer, J.L. da Silva Jr. and D. Patel, "PicoRadio: Ad-Hoc Wireless Networking of Ubiquitous Low-Energy Sensor/Monitor Nodes ," *Proceedings of the IEEE Computer Society Annual Workshop on VLSI (WVLSI'00)*, Orlando, Florida, April 2000.

[23] S. Singh, C.S. Raghavendra, "Power efficient MAC protocol for multihop radio networks," *Proceedings PIMRC'98*, Sept. 1998.

[24] S. Singh, M. Woo, C.S. Raghavendra, "Power-aware routing in mobile ad-hoc networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MOBICOM'98)*, pages 76-84, October 1999, Dallas, Texas.

[25] J. Warrior, "Smart Sensor Networks of the Future," *Sensors Magazine*, March 1997.

[26] A. Woo and D.E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom '01)*, July 2001, Rome, Italy.

[27] K.Chakrabarty, S. S. Iyengar, H. Qi, E.C.Cho, " Grid Coverage of Surveillance and Target location in Distributed Sensor Networks" *To appear in IEEE Transaction on Computers*, May 2002.

[28] S.S.Iyengar,M.B.Sharma,and R.L.Kashyap,"Information Routing and Reliability Issues in Distributed Sensor Networks" *IEEE Tran. on Signal Processing*, Vol.40, No.2, pp3012-3021, Dec. 1992.

[29] S. S. Iyengar, D. N. Jayasimha, D. Nadig, "A Versatile Architecture for the Distributed Sensor Integration Problem," *IEEE Transactions on Computers,* Vol. 43, No. 2, February 1994.

[30] L. Prasad, S. S. Iyengar, R. L. Rao, and R. L. Kashyap, "Fault-tolerant sensor integration using multiresolution decomposition," *Physical Review E*, Vol. 49, No. 4, April 1994.

[31] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16-27, October 2000.

[32] G. Asada *et al.*, "Wireless Integrated Network Sensors: Low Power Systems on a Chip," *Proceedings of the 1998 European Solid State Circuits Conference.*

[33] M. Chu, H. Haussecker, F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks." *International Journal on High Performance Computing Applications*, to appear, 2002.

[34] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri, "Querying the Physical World, " *IEEE Personal Communications*, Vol. 7, No. 5, October 2000.

[35] B. Krishnamachari, R. Bejar, and S. B. Wicker, "Distributed Problem Solving and the Boundaries of Self-Configuration in Multi-hop Wireless Networks" Hawaii International Conference on System Sciences (HICSS-35), January 2002.

[36] R. Min *et al.*, "An Architecture for a Power-Aware Distributed Microsensor Node", *IEEE Workshop on Signal Processing Systems (SiPS '00)*, October 2000.

[37] P. Gupta and P. R. Kumar, Critical power for asymptotic connectivity in wireless networks. *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W.M. McEneaney, G. Yin, and Q. Zhang (Eds.)*, Birkhauser, Boston, 1998.",

[38] B. Krishnamachari, R. Bejar, and S. B. Wicker, "Phase Transition Phenomena in Wireless Ad-Hoc Networks," *Symposium on Ad-Hoc Wireless Networks, Globecom 2001*, 2001.

[39] R. Wattenhofer, L. Li, P. Bahl, and Y. M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," *IEEE Infocom*, 2001.

[40] T.-C. Hou and V. O. K. Li, "Transmission range control in multihop radio networks," *IEEE Transactions on Communications*, vol. 34, no. 1, pp. 38–44, January 1986.

[41] R. Ramanathan, and R. Hain, "Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment," *INFOCOM 2000*.

[42] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar, "Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW protocol," *Proceedings of European Wireless 2002. Next Generation Wireless Networks: Technologies, Protocols, Services and Applications*, pp. 156-162, Feb. 25-28, 2002.

[43] H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals," *IEEE Trans. On Communications*, vol. COM-32, pp. 246-257 (March 1984).

[44] S.Chessa, P.Santi, "Crash Faults Identification in Wireless Sensor Networks," to appear in *Computer Communications*, Vol. 25, No. 14, pp. 1273-1282, Sept. 2002.

[45] D. Estrin *et al. Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*, National Research Council Report, 2001.

[46] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," *INFOCOM 2002*, New York, NY, USA, June, 2002.

[47] Nirupama Bulusu, John Heidemann, and Deborah Estrin, "GPS-less Low Cost Outdoor Localization For Very Small Devices," *IEEE Personal Communications Magazine*, 7 (5), pp. 28-34, October, 2000

[48] R. A. Maxion, "Toward diagnosis as an emergent behavior in a network ecosystem," in *Emergent Computation*, Ed. S. Forrest, MIT Press, 1991.

[49] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *INFOCOM* 1997.

[50] N. Megiddo, Linear time algorithm for linear programming in R3 and related problems, SIAM J. Comput. 12(4) (1983) 759-776.

[51] M.E. Dyer, Linear time algorithms for two and three-variable linear programs, SIAM J. Comput. 13(1) (1984) 31-45.

[52] B. Cain, T. Speakman, D. Towsley, "Generic Router Assist (GRA) Building Block Motivation and Architecture," *RMT Working Group, Internet-Draft <draft-ietf-rmt-gra-arch-01.txt>*, work in progress, March 2000.

[53] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80-86, January 1997.

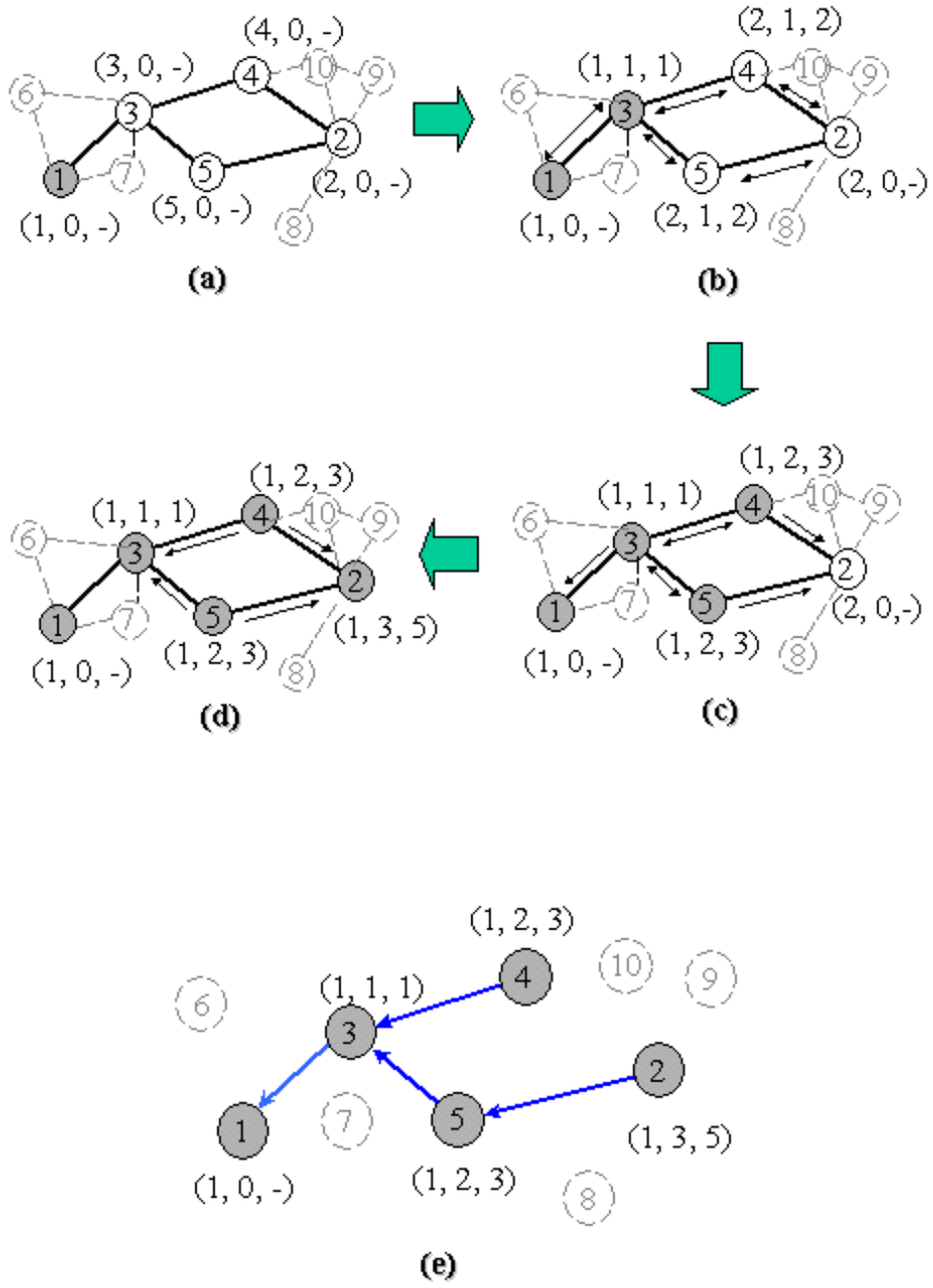

Figure 10: An illustration of the proposed algorithm for feature cluster formation. (a)-(d) show the message exchanges and table updates during the algorithm, and (e) shows the final feature-cluster that is formed.

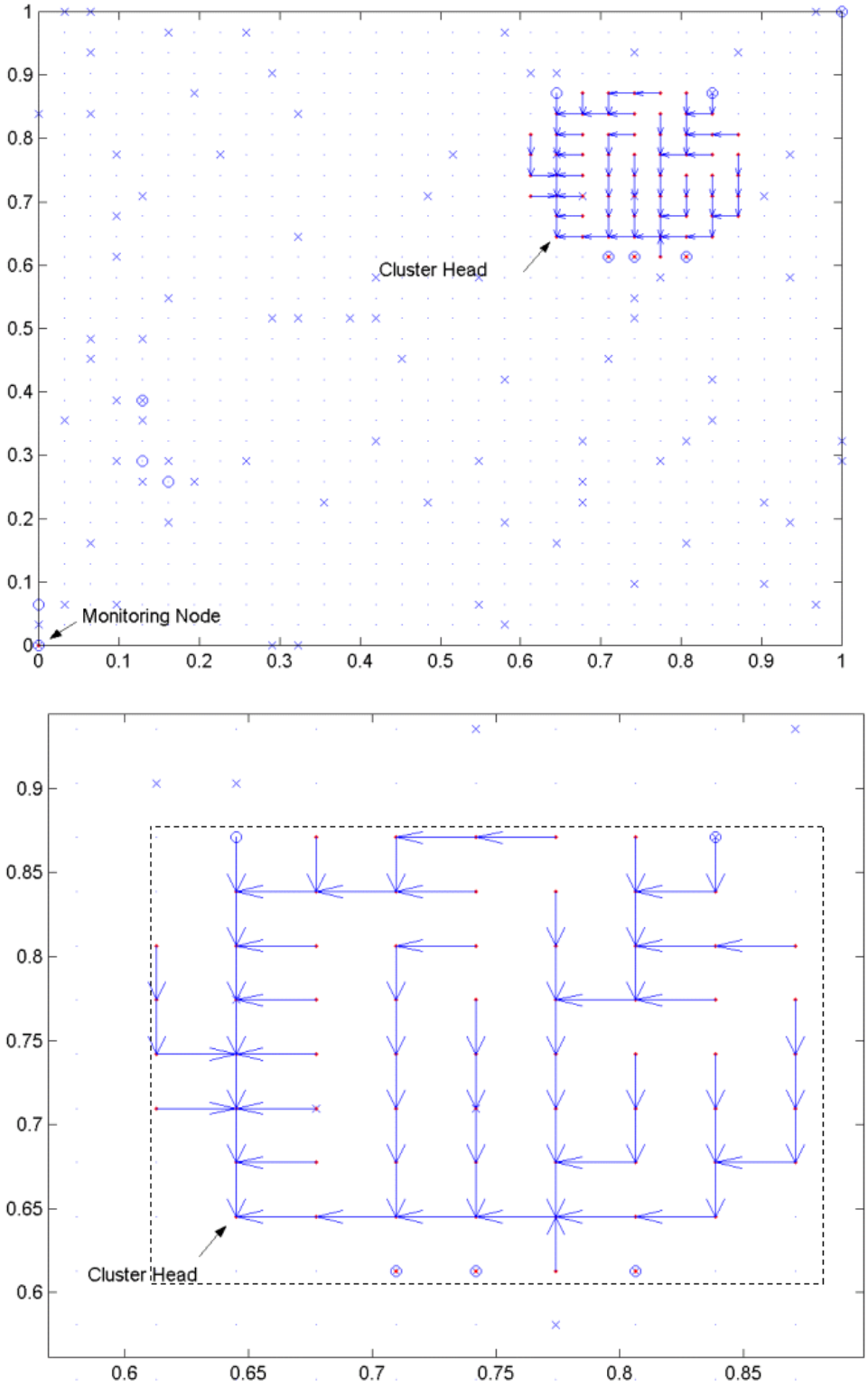

Figure 11: A simulator snapshot depicting the intra-cluster routing tree within the feature region along with the elected cluster head. The dashed rectangle on the bottom represents the rectangular approximation of the feature region that can be represented compactly. Note that the rectangular approximation is highly robust to faults and decoding errors of nodes on the border of the feature region.

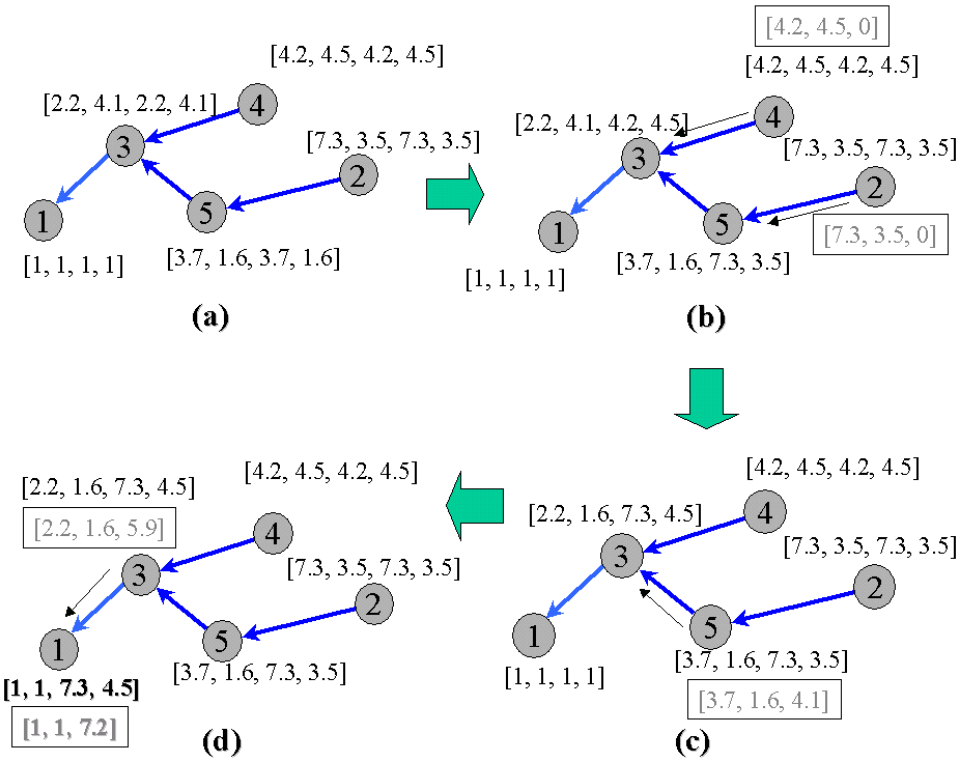

Figure 12: Illustration of the stepwise rectangular approximate aggregation (SRA) scheme. The 4-tuples represent the lower-left and upper-right x and y coordinates for the corresponding rectangle. The 3-tuples are the $[XMIN, YMIN, DIAG]$ representations that are communicated at each transmission upstream.