TK
7872
.D48
J34
2006

Event Boundary Detection Using Autonomous Agents in a Sensor Network

By

Adil Jaffer

A project
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of

Master of Engineering

in the department of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2006

Author's Declaration:

I hereby declare that I am the sole author of this project.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

[illegible]

Adil Jaffer

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

[illegible]

Adil Jaffer

Borrow List

Acknowledgements

I would like to thank my advisor Dr. Jaseemuddin for his support with the research and work that has gone into this project. I'd also like to thank my family and especially my wife for all the support they've provided.

Abstract

We propose a novel approach to event boundary detection, where autonomous agents are deployed in order to minimize the number of transmissions required to discover an event boundary. The goal of our algorithm is to reduce the number of non-boundary node transmissions (i.e. nodes within the event area and not within transmission distance to the boundary), since the sensory data from these nodes are not required for event boundary detection. The algorithm works by first randomly generating a fraction of agents within the event nodes, then discovering and mapping the boundary, and finally reporting the aggregated results to the user. Simulations demonstrate that the algorithm exhibits O(n) efficiency relationship with the event area, which is an improvement over existing methods that show $O(n^2)$ relationships. Furthermore, we demonstrate that the boundary of an event may be successfully mapped using the proposed algorithm.

Table of Contents

List of Figures

List of Abbreviations

WSN	Wireless Sensor Network
LEATCH	Low Energy Adaptive Clustering Hierarchical protocol
TEEN	Threshold sensitive Energy Efficient sensor Network protocol
ATEEN	Adaptive Threshold sensitive Energy Efficient sensor Network protocol
SPIN	Sensor Protocols for information via Negotiation protocol

Chapter 1: Introduction

With the advent of wireless communications and the ever decreasing cost to performance ratio of microcontrollers; smart, low-cost wireless sensor networks have gained feasibility. A sensor network consists of a set of wireless processing nodes spread over a region of space in order to provide spatial resolution to the sensory data from that region. This network could be seen as a mesh of interconnected smart nodes, each with the capability of relaying sensory data across multiple hops until a map of sensory data is finally delivered to the user. A seismological application could be taken as an example of such a scheme, where sensor nodes could be spread over a region susceptible to earthquakes. A seismologist could then learn the varying degrees of vibration seen throughout the entire region, providing useful information such as the location of the epicenter or even the wave propagation characteristics of the earthquake.

The main constraint for realizing a practical sensor network is cost; which in turn can be broken down into the cost of the individual node as well as the implementation of the overall network. The cost of a node can be reduced by decreasing the size and complexity of the node. Implementation costs can be reduced by removing restrictions to node placement as well as the need for infrastructure. Without infrastructure the sensor node becomes completely wireless and ad-hoc, with limited power resources and the need for wireless communications. It is this need for power efficiency that drives much of the current work in wireless sensor networks (WSNs), where efficiency is improved through the use of various algorithms that balance the trade-off between sensor resolution and cost/efficiency.

In many typical wireless sensor network applications scalability is a difficult issue to address. If we assume that a given network is spread over a considerable distance then the number of nodes which must report their sensory data increases accordingly. Therefore, as a sensor network becomes larger, the task of routing information to the user efficiently becomes increasingly more difficult.

It is with this consideration that various wireless sensor network protocols have been developed, and with the intent on reducing power consumption through the reduction in the overall number of transmissions without sacrificing the application requirements. Typically, these methods involve either the restriction of transmissions to only those nodes that posses' relevant data or through the aggregation of data within the network before it reaches the user. Both of these methods require a data centric approach where the user communicates to the network which types of data and what value ranges are of interest.

For the case of boundary detection of large event areas the current algorithms are not optimal. If we assume that the information of interest is the event boundary and not the individual sensor readings within the event region, then it becomes a waste of network resources to report the sensor readings of all nodes in the event region. Additionally, even if only the nodes along the event boundary report their sensor data then there is the added overhead of passing that event data through the sensor network in order to reach the user. Ideally, an algorithm should take into account the event boundary path and utilize that path to aggregate the event boundary data and reduce overall number of nodes reporting to the sink.

Chapter 2: Background

In this chapter we discussed recently proposed routing protocols and event detection schemes for WSN, which are relevant to our work.

2.1 Routing Techniques

Routing protocols for WSN can be broadly categorized into Hierarchical and Data Centric routing. We first discuss the Hierarchical routing protocols in section 2.1.1 followed by Data Centric protocols in section 2.1.2.

2.1.1 Hierarchical Routing Protocols

The main goal of a hierarchical protocol is to reduce power consumption through the use of clustering. A cluster is formed within a localized region, where transmission distance will not deplete resources as much. All queries occur between the sink (or an intermediary node) and a single node in the cluster (usually termed cluster leader). Once all nodes within a cluster have communicated their sensor data, the cluster leader then responds to the query. The main advantage of clustering is that it allows for scalability since the higher echelon of nodes querying the cluster leaders do not require knowledge of the cluster member nodes. Additionally, since clusters are localized regions data aggregation is facilitated.

The LEACH (Low Energy Adaptive Clustering Hierarchical protocol) protocol was one of the earliest attempts at a hierarchical protocol. In this protocol Clusters are formed around cluster heads based on signal strength. It is important to not that nodes are not selected as cluster heads rather they elect themselves randomly. Once a node volunteers to be a cluster node it then advertises its services to its neighbors who in turn decide to join the cluster based on signal strength. All queries to and from nodes within the cluster are routed through the cluster head which performs the task of data aggregation and routing back to the sink. Once a fixed time elapses new cluster heads are chosen in order to distribute overhead. The main drawback behind the LEACH protocol is that it assumes that all nodes can communicate directly to the sink. This is not always possible in larger networks and usually requires more transmission power then multi-hop routing. Additionally, in order to setup clusters and cluster heads, a lot of network overhead is required. Finally, the use of randomly elected cluster heads may lead to "holes" in the network where some nodes are not sufficiently near enough to a cluster head to become a member of a cluster.

The TEEN protocol (Threshold sensitive Energy Efficient sensor Network protocol) is a hierarchical protocol that uses a data centric approach to improve energy conservation [13]. In TEEN clusters are formed on multiple levels, where one cluster heads form their own clusters. These clusters form a tree structure originating from the sink. Once this tree structure is formed the sink sends a hard threshold and a soft threshold to all nodes in the network. These thresholds are used by the sensing nodes to report back to the sink, up the cluster tree, any sensed data which passes those thresholds. Once a sensing node detects data above the hard threshold it sends that data back through the network. However, it will not resend that sensed data until the sensed data has changed enough to pass the soft threshold. However, this leads to the main disadvantage of the TEEN protocol since the network will fall silent if no thresholds are reached. Additionally, the network has no knowledge if nodes have received hard or soft thresholds since some nodes are expected to remain silent if thresholds are not reached.

The ATEEN protocol (Adaptive Threshold sensitive Energy Efficient sensor Network protocol) is an extension of TEEN and introduces periodic reporting [14]. In the ATEEN protocol, a reporting schedule is handed out to all nodes when the soft and hard thresholds are transmitted. Nodes are given a transmission slot which they must transmit their status regardless of thresholds. This leads to more energy consumed, but a more responsive and robust protocol. The main drawbacks for both TEEN and ATEEN is the added complexity of setting up multiple cluster heads at various levels as well as maintaining thresholds and reporting schedules (in the case of ATEEN).

2.1.2 Data Centric Routing Protocols

Typical routing protocols operate on the principle that the user requests data from the network, that request is routed throughout the network, and nodes with valid data respond to that request. This paradigm is useful in scenarios where the user makes infrequent requests and does not need to be continuously informed of events sensed out in the network. For these cases data centric routing was formulated where the number of events requiring observation or few and far between. Data centric routing formulates that the network should be given knowledge of what types of data is of interest by the user and then left to themselves to report if they have seen that data. This requires for the network to have some application knowledge as well as a naming convention for the data.

The SPIN protocol (Sensor Protocols for information via Negotiation protocol) was one of the first data centric protocols proposed [12]. This protocol uses a meta-data descriptor to describe their sensed data. This meta-data descriptor is much smaller then the actual data itself but is useful in order to communicate to other nodes what data that node posses. Once new data and a new meta-data descriptor are available that descriptor is advertised to the nodes neighbors. If those nodes have not themselves collected similar data then they will already have that meta-data descriptor and will ignore the message. If neighbors do not have the meta-data descriptor they will then store it and forward it themselves to their neighbors. In this way the meta-data descriptor for an event can be propagated until the entire network has knowledge of that event. The user may then poll any node in the network in order to retrieve the meta-data descriptors which then point to the source nodes. The use of negotiating with meta-data descriptors avoids flooding the network and therefore is highly energy efficient. However the main drawback behind spin is that there is no guarantee for data delivery since not all nodes may be interested in the data between source and destination nodes.

The directed diffusion protocol is a very popular data centric protocol that has gained much attention in this field [11]. The main idea behind directed diffusion is that when an event occurs, a local gradient (or information flow) is setup within the local area of the event nodes. The requester then diffuses interest throughout the network which when encountering nodes that posses the required data, setup additional gradients which flow data back towards the sink. In this method multiple paths are formed from the same sources and therefore only the strongest paths are reinforced while the weaker paths are let go. A stronger path is defined as one where a lot of data aggregation is occurring over a weaker path with less data aggregation. The selection of various paths over others is primarily used in order to improve data aggregation however it may also be used to improve reliability and robustness. An important addition to direct diffusion is the use of caching, which allows for an interest to be stored for later use.

Another version of directed diffusion is gradient routing, which utilizes the minimum number of hops to the sink rather, then data aggregation in order to determine which paths survive. Each node keeps track of the height to the sink or the number of hops to the sink. The difference between two nodes heights is referred as the gradient and the sharpest gradient is used to route the packet back towards the sink.

The main advantages behind direct diffusion are the lack of any overall routing topology. In directed diffusion a node only needs knowledge of 1-hop neighbors rather then constructed routing topologies. This reduces overhead required to rebuild any such topologies whenever there is a topology change. Additionally, directed diffusion is highly energy efficient since it is an ideal data aggregation model with only 1-hop data transmissions. The main disadvantage behind directed diffusion is that it is application dependent and must be set-up that way from the start.

2.2 Event Boundary Detection

Chintalapud [1] laid the groundwork for localized edge detection, where he proposed in his paper three approaches to event boundary detection. Each of the proposed methods restricted the number of nodes reporting to the user to only the nodes that have self-determined themselves as boundary nodes. Nodes determine their status by retrieving sensory data from all neighboring nodes and performing 1 of 3 suggested algorithms to self determine their boundary status. These three algorithms are:

A statistical approach where each node in the event area would compile a statistic based on the results obtained from all of its neighboring nodes. The static is then compared to a Boolean decision function. This method has the advantage that it is robust to errors however it does not take into account the geographical location of each node, resulting in boundary estimation errors.

An imaging approach facilitated through the use of edge filters such as the Prewitt filter. This method utilizes a weighted approach to overcome the Prewitt filters need for regularized pixel like data. The Prewitt filter rotates an edge template around the 8 neighboring regions of a pixel in order to determine the closest match to a given edge orientation. Since node placement is random and any given sub-region within the node area may have more neighboring nodes then another region; biasing can occur if the geographic location is not taken into account. Therefore weights are applied to each node corresponding to its location relative to specific regions.

Finally the he proposes the use of a linear classifier to determine which nodes are edge nodes. The linear classifier optimally places a linear line such that the maximum number of non-event nodes lies to one side while the maximum number of event nodes lies on the opposite side. If the node (i.e. the epicenter and a given radius r from the epicenter) is on the side of the edge nodes it is deemed an edge node. This method has the advantage that it inherently takes into account the geography of nodal placement

Finally, once all nodes that have witnessed the event have determined their status, all nodes which are within the event boundary report back to the sink.

There are several disadvantages to all three methods. The first disadvantage is that only local area neighborhood data is considered when evaluating an edge condition. Nodes that exist along network holes may not be considered since although they border the edge of the event, there may not be a neighboring non-event node within transmission radius. This may lead to event boundaries that are not contiguous. Additionally, all nodes within the event area must poll their neighbors to discover if they are an edge node or not. Assuming that a completely efficient algorithm is used; all nodes within this region must transmit at least once. With a constant node density, as an event area increases in size the corresponding number of internal nodes increase exponentially. Therefore for larger event areas the algorithm becomes increasingly resource intensive.

Further work in using local statistics of the node neighborhood to discover event boundaries have been presented by both [4] and [8]. In these papers the authors utilize local neighborhood sensor data to determine the likelihood of a node being an edge (a classifier is used in [4] while a statistical approach is used in [8]. The main addition of these two newer algorithms to the literature is the use of neighboring information to exclude faulty nodes. These algorithms are still subjected to the same disadvantage, where all nodes within the event area must discover neighboring node information.

An event boundary detection algorithm involving hierarchical sensor networks was proposed in [2] where the authors proposed using a cluster based approach. Initially, the cluster formation occurs similar to that done in hierarchical algorithms such as LEATCH or (A)TEEN. The sensor network is then divided into quadrant where a single cluster head in each quadrant is then polled for event status. This process continues as each cluster head within each quadrant further sub-divides its region into four sub-quadrants and each sub-quadrant further divides until maximum resolution is reached. The final sub-quadrant's that contain event boundary information are then relayed back to the sink.

Additional work on event boundary detection in hierarchical networks has been done by [4], where cluster heads are chosen as local maxima or minima in order to allow for contour lines to be discovered through the polling of bisecting nodes. The method follows as: cluster formation; course and fine localization; cluster reorganization; and contour line construction. Cluster heads begin by polling member nodes and each head determines if its region contains local maxima/minima (the maxima/minima points correspond to sensory data peaks within the event region). Clusters then re-organize around cluster heads with maxima/minima points, merging clusters and electing new cluster heads. Within each newly reorganized cluster, new sub clusters are formed at the maxima/minima and they poll selected cluster nodes along radiating paths. Since these radiating paths would bisect the contour lines of the maxima/minima the sensor results of the nodes along those paths allow for the calculation of contour line points. The cluster hierarchy then reports the contour lines back to the sink.

The main disadvantage of these methods is that although they spatially separate sampling of the event area in order to reduce the total number of nodes polled they still operate on the hierarchical polling method which requires high maintenance overhead, long transmission distances and a two-tiered node structure. Additionally, this method works on a polling principle where the user must poll the network to discover the event boundary.

2.3 Terminology

In this section we defined the terminology used in this paper. A sensor network consists of a multitude of wireless sensors spread over an area of interest. These wireless sensors would be interconnected wirelessly, limited by local transmission distances. This would lead to a mesh like structure, where each node would have a communication link with its immediate neighbor and would ideally use multi-hop transmissions to communicate with the user.

If we assume that there exists a case example of a sensor network (**N**) which was the set of sensor network nodes from $\mathbf{N_1}$, $\mathbf{N_2}$, ... ,$\mathbf{N_n}$ where **n** is the maximum number of nodes.

Furthermore, let us assume that the sensor network **N** detects a homogenous and contiguous event, and that event encompasses an event region defined by a subset of **N** to be refereed to as $\mathbf{N_{event}}$, where $\mathbf{N_{event}}$ contained a number of nodes denoted by $\mathbf{n_{event}}$. If the sensor network had a constant density (ρ) and event radius ($\mathbf{R_{event}}$) then the number of nodes within $\mathbf{N_{event}}$ would be defined as

$$\mathbf{n_{event}} = \pi \bullet \mathbf{R_{event}}^2 \bullet \rho \qquad \text{(eq. 1)}$$

Furthermore it can be seen from eq.1 that if the sensor network had a constant ρ then if $\mathbf{R_{event}}$ was increasing linearly then $\mathbf{n_{event}}$ would increase exponentially. Additionally, it can be seen that if we assume all nodes have a transmission radius ($\mathbf{R_{TX}}$) smaller then $\mathbf{R_{event}}$ then there would be a subset of **N** capable of detecting the event boundary, denoted by $\mathbf{N_{all\_boundary}}$ and consisting of $\mathbf{n_{all\_boundary}}$ number of nodes. $\mathbf{N_{all\_boundary}}$ is dependant on $\mathbf{R_{TX}}$, where

$$\mathbf{n_{all\_Boundary}} = \pi \bullet \rho \bullet [(\mathbf{R_{event}} + \mathbf{R_{TX}})^2 - (\mathbf{R_{event}} - \mathbf{R_{TX}})^2] \qquad \text{(eq. 2)}$$

or

$$\mathbf{n_{all\_Boundary}} = \pi \bullet \rho \bullet 4(\mathbf{R_{event}} \bullet \mathbf{R_{TX}}) \qquad \text{(eq. 3)}$$

Therefore it is easy to see that $\mathbf{n}_{\mathbf{event}}$ has an exponential relationship to $\mathbf{R}_{\mathbf{event}}$, while $\mathbf{n}_{\mathbf{all\_boundary}}$ has a linear relationship to $\mathbf{R}_{\mathbf{event}}$.

Furthermore $\mathbf{N}_{\mathbf{all\_boundary}}$ would consist of the subset of all boundary nodes, which are both within the event and within the boundary region ($\mathbf{N}_{\mathbf{event\_boundary}}$) as well as the subset of nodes, which are only within the boundary region and not within the event ($\mathbf{N}_{\mathbf{non\_event\_boundary}}$). $\mathbf{n}_{\mathbf{non\_event\_boundary}}$ is given by:

$$\mathbf{n}_{\mathbf{non\_event\_boundary}} = \pi \bullet \rho[(\mathbf{R}_{\mathbf{event}} + \mathbf{R}_{\mathrm{TX}})^2 - (\mathbf{R}_{\mathbf{event}})^2] = \pi \bullet \rho\,(2\mathbf{R}_{\mathbf{event}} \bullet \mathbf{R}_{\mathrm{TX}} + \mathbf{R}_{\mathrm{TX}}^2) \qquad \text{(eq. 4)}$$

While $\mathbf{N}_{\mathbf{event\_boundary}}$ is given by:

$$\mathbf{n}_{\mathbf{event\_boundary}} = \pi \bullet \rho[(\mathbf{R}_{\mathbf{event}})^2 - (\mathbf{R}_{\mathbf{event}} - \mathbf{R}_{\mathrm{TX}})^2] = \pi \bullet \rho(2\mathbf{R}_{\mathbf{event}} \bullet \mathbf{R}_{\mathrm{TX}} - \mathbf{R}_{\mathrm{TX}}^2) \qquad \text{(eq. 5)}$$

Leading to the conclusion that the difference between $\mathbf{n}_{\mathbf{non\_event\_boundary}}$ and $\mathbf{n}_{\mathbf{event\_boundary}}$ would consist of $\pi \bullet \rho \bullet 2 \bullet RTX^2$. If we assume R_{TX} is constant then as $\mathbf{R}_{\mathbf{event}}$ increases the assumption that $\mathbf{n}_{\mathbf{non\_event\_boundary}} \approx \mathbf{n}_{\mathbf{event\_boundary}}$ becomes true.

It must be noted that for the above calculations to be valid, the assumption that the event can be characterized fully by a radius must be valid as well. As the event region becomes more and more irregular, the ratio of perimeter length versus event area becomes larger and therefore $\mathbf{n}_{\mathbf{event\_boundary}} \rightarrow \mathbf{n}_{\mathbf{event}}$. Considering that most applications requiring event region characterization such as temperature, acoustics, chemical analysis and vibration propagate in an omni-directional manner due to either diffusion or the wave-like nature of energy, it is safe to assume that many applications will be contiguous and mostly spherical in nature. Therefore the assumption that the event can be characterized by a radius will hold true in many applications leading to the conclusion that the above projections will hold.

Chapter 3: Our Event Detection Algorithm

We propose a novel algorithm of event boundary detection where the event region boundary is mapped and reported by autonomous agents. The main objective of this algorithm is to improve the transmission efficiency of event boundary detection.

The first technique we use to improve efficiency is through decoupling the increase of efficiency from the increase in event area ($\mathbf{T_{total}} \propto \mathbf{R_{event}}^2$) which is an exponential relationship to event radius and move towards an algorithm that has an efficiency increase related to the event circumference ($\mathbf{T_{total}} \propto \mathbf{R_{event}}$). It is clear that as an event area increases in size (within a network with constant node density) that the total number of nodes that see the event will increase exponentially when compared to the number of nodes that will see the event boundary. Since nodes have no knowledge of whether they exist on a boundary until after they have polled their local neighborhood there is no way to adapt previous methods mentioned in the literature to reduce interior node transmissions. By first locating the event boundary region and then propagating event boundary discovery transmissions along that boundary, we hope to reduce interior node transmissions.

Additionally, it is more beneficial to report only absolute boundary nodes rather then all boundary nodes. All boundary nodes refer to all nodes within R_{TX} distance of the actual event boundary. Absolute boundary nodes can be defined as the minimum set of nodes, which are required to completely circumscribe all event nodes. It is apparent that absolute boundary nodes ($\mathbf{N_{absolute\_boundary}}$) are a subset of the event boundary nodes ($\mathbf{N_{event\_boundary}}$) and therefore $\mathbf{n_{absolute\_boundary}} << \mathbf{n_{event\_boundary}}$. If the application requires only the event area shape and not the event boundary data statistics (which by the statistics very nature will be similar data sets divergent only by the tolerance of the boundary function) then the absolute boundary nodes are more then sufficient and the additional transmissions created by polling all boundary nodes as well as the larger packet size that is required to report the extraneous data back to the sink are wasted. The proposed algorithm conserves this energy by minimizing polling of non-event boundary nodes by only requesting event boundary nodes to respond to a query and by assuming the state of non-event boundary nodes through their null response.

Finally, event boundary detection efficiency can be improved through the utilization of node collaboration through roaming agents. Agents are program code, which run locally on the individual nodes. Once the agent code has completed, the results of the program as well as the program code are compiled in a packet and sent to an adjacent node. The adjacent node could then open the packet and run the agent code, utilizing the compiled results from the previous node (also included in the packet) as an input. The agent would then proceed to create a new result, based on the co-processing of both nodes. This process could then continue, as the agent is passed from node to node throughout the network. An example of an agent would be a program which would compile the average sensor readings of all nodes along the agent path.

Agents are autonomous in the sense that once created they do not require input from the user to perform their function. In this algorithm we propose to use agents in order to discover an event boundary node and then map the entire event boundary by moving along linkages between the absolute event boundary nodes. The function of the agent would be two-fold: record its path along the boundary of the event as well as utilize neighboring information to determine which neighboring node is the next absolute event boundary node. Utilizing agents reduces the total number of nodes needing to report to the sink, without the need to resort to clustering.

Additionally, the use of agents allows for the selective polling of neighbors, since if the agent knows it's on the event boundary, it can use that knowledge to selectively poll nodes which have either detected the event or have not detected the event. This allows for the algorithm to reduce the overall polling of boundary nodes by limiting itself to polling only event-boundary nodes and not non-event-boundary nodes. Since $N_{event\_boundary} \approx N_{non\_event\_boundary}$ (for a sufficiently large network) the total number of transmissions required can be reduced by at most 50%.

3.1 Assumptions

We have made several assumptions in order to facilitate the simulation of the proposed algorithm in a sensor network using Matlab. The first assumption is that there exists a wireless mesh network of autonomous sensor nodes spread over a geological area. The distribution of those nodes would be random, with a fixed overall network density. Additionally, the nodes would have a finite amount of energy available and therefore energy conservation would be of concern. A MAC layer protocol would also exist which would allow for the individual nodes to transmit and receive messages between neighbors within a given transmission radius. Additionally, the MAC layer protocol would avoid collisions either through the use of scheduling or a back-off algorithm and therefore the simulation will not consider collisions. Finally we have assumed that each node will know its local co-ordinate relative to the sink as well as knowledge of its neighboring node locations. The assumption that there exists a protocol layer sufficient to support the above mentioned requirements is reasonable since much previous work in sensor networks has focused on these areas.

We have also assumed that the sensor network will have the capability to run an agent program transmitted to the node from another neighboring node. The agent located locally on the node would have access to all incoming and outgoing transmissions as well the capability to store data locally on the node, to be recovered later by a subsequent agent.

Additionally we have assumed that events detected by the network are such that they exist as a contiguous region, with a roughly circular shape. Finally, since errors in sensor measurements are beyond the scope of this paper we do not considered them during simulation.

3.2 Summary of the algorithm

The algorithm involves several distinct phases of operation:

1) A node detects an event within a preset time period/window. Once the time period has ended the nodes that detected the event will randomly generate an agent based on a preset threshold probability. This allows for a small percentage of agents to be generated.

2) Once an agent is first created it defines a random direction vector and moves in that direction to discover the edge of the event region. It knows that it has reached the event region once it hears a response from a node that has not seen the event. If the agent discovers another agent has visited this node then it knows that another agent exists which can complete the task and therefore ceases operation. In order to reduce transmissions, during the edge discovery phase, the agent requests responses from only nodes that have not seen the event.

3) When an agent has found the boundary, a second agent (or child agent) is spawned.

4) Both agents move along the boundary (parent agent in CW fashion while child agent CCW). As the agent reaches the next node it:
   I. Retrieves event status of all neighboring nodes
   II. Sorts' nodes based on direction angle relative to the angle from the previous node in the agents path, and move to the next event node on that sorted list.
   III. Stores all nodes that agent has visited that are on the boundary in the agent's boundary stack
5) When a parent agent meets the path of a child agent, the node hosting the agent combines the stored agent paths and reports to the sink.

The basic operation of this algorithm requires that the detection of an event be separated within individual periods or time windows. The period would define the frequency in which the algorithm would run as well as the time resolution of the resulting data. Once a period of time has passed, each node that detected the event within the region can then generate an agent. Since this algorithm would operate ideally with only one agent generated, a random probability is used to reduce the total number of event nodes that generate an agent. Therefore each node that wishes to create an agent must first randomly generate a number and if that number is below a pre-set threshold the agent will be generated. The wake-up threshold can be chosen based on the allowable resolution of the event area, since the smaller the event area the fewer nodes exist and therefore the less likely a small threshold will generate an agent.

Once the agents are generated they then begin searching for the boundary of the event area. This is accomplished through performing a directed random walk across the network until a non-event boundary node is reached. The random walk consists of first choosing a direction vector and then moving along that vector until a boundary node is reached. Since there is a higher probability of the agent originating from an internal node, only nodes that have not seen the event need to reply to the node hosting the agent. This reduces the overall number of transmissions required to find the boundary. During this random walk, if the agent encounters the path of another agent (one that is either searching for the boundary or is moving along the boundary) it ceases. Again, since the ideal algorithm would only generate one agent there is no need for the agent to continue since there is already an existing agent running.

When the agent discovers a boundary node it then begins moving along the boundary. This involves first generating a child agent which then moves along the opposite direction along the boundary, away from the parent agents' direction. This reduces the time required to detect an event region since there are two simultaneous agents moving along the boundary rather then a single agent. Ideally only one agent would be generated, which would then find the boundary and split, moving along the boundary in opposite directions. Once both parent and child agent meet, the child agent passes along its boundary path to the parent agent, who would in-turn pass it along to the sink.

In order to discover the next boundary node (**node$_{n+1}$**) the agent must first learn which of its neighbors have seen the event and which have not. In order to discover the event status of its neighbors the node hosting the agent (**node$_n$**) polls all of its neighbors requesting for a response from only those who have not seen the event. Once this information is gathered the node hosting the agent then uses it's knowledge of the location of it's neighbors as a pre-generated table of angles to determine which event node lies clock-wise (or counter-clockwise depending on the direction the agent is traveling on the boundary) from the previous node in the agent's path (**node$_{n-1}$**). It is clear that if an imaginary line is drawn from **node$_{n-1}$** to **node$_n$** and if that line was to move clockwise (with **node$_n$** as its pivot) until it encountered a node that had seen the event then that node would be the next node along the boundary path (**node$_{n+1}$**). This case is true for the majority of boundary decisions made by the agent except for a couple of specific scenarios that will be detailed in section 3.3.

3.3 Transmission Cost

In order to calculate the cost associated with a statistical method (the control which the algorithm we are proposing must be compared to) as proposed in [1] we must take into account that all nodes within the event as well as all nodes within $\mathbf{R_{TX}}$ of any node within the event must exchange their event states. This can be characterized by:

$$\mathbf{n_{TX\_control}} = \pi \bullet \rho(\mathbf{R_{event}} + \mathbf{R_{TX}})^2 \qquad \text{(eq. 6)}$$

The algorithm that we have proposed will have the following transmission cost. Once the algorithm has completed, the number of nodes related to the wake-up threshold ($\mathbf{TH_{wakup}}$) will have created an agent. These agents will have traversed an average distance of $\mathbf{R_{event}}$ to reach the event boundary. Additionally, the average distance between nodes will be half the transmission radius if we assume an even distribution of nodes. Therefore the total number of node transmissions required to at most:

$$\begin{aligned}\mathbf{n_{TX\_boundary\_discovery}} &= 2 \bullet (\mathbf{R_{event}}/\mathbf{R_{TX}}) \bullet \mathbf{n_{event}} \bullet \mathbf{TH_{wakup}}. \qquad \text{(eq. 7)}\\ &= 2 \bullet (\mathbf{R_{event}}/\mathbf{R_{TX}})(\pi \bullet \mathbf{R_{event}}^2 \bullet \rho) \bullet \mathbf{TH_{wakup}}\end{aligned}$$

As long as $(\mathbf{R_{TX}}/\mathbf{R_{event}}) << 1$ and $\mathrm{TH_{wakup}} << 1$ is true then the number of interior event nodes polled by the algorithm will be a fraction of the total number of interior nodes. Additionally, since during the boundary discovery phase of the algorithm only non-event boundary nodes are polled, the total number of boundary nodes will be reduced by almost 50% considering that $\mathbf{N_{non\_event\_boundary}} \approx \mathbf{N_{event\_boundary}}$ as pre equation 5. Therefore the total number of nodes which must be polled along the boundary will be equal to $\mathbf{n_{event\_boundary}}$ as well as the cost to transmit the agent information between nodes.

$$\mathbf{n_{TX\_agent\_overhead}} = 2 \bullet \pi \bullet \mathbf{R_{event}} \bullet (2/\mathbf{R_{TX}}) \qquad \text{(eq. 8)}$$

Therefore the total cost of the algorithm would be equivalent to:

$$n_{TX\_proposed} = n_{TX\_boundary\_discovery} + n_{TX\_agent\_overhead} + n_{event\_boundary} \quad \text{(eq. 9)}$$
$$= 2 \bullet (R_{event}/R_{TX})(\pi \bullet R_{event}^2 \bullet \rho) \bullet TH_{wakup} + 2 \bullet \pi \bullet R_{event} \bullet (2/R_{TX}) +$$
$$+ \pi \bullet \rho(2R_{event} \bullet R_{TX} - R_{TX}^2)$$

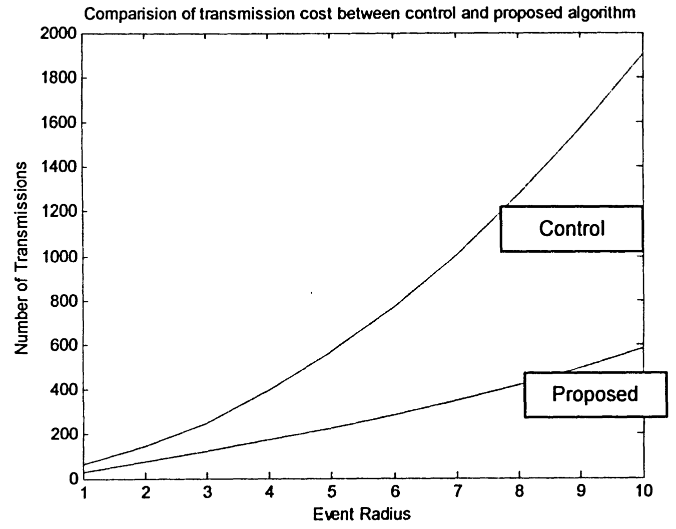

Figure 1: graphical results of the transmission cost of the control algorithm and the proposed algorithm.

3.4 Special cases for determining next node in boundary path

During the event boundary discovery phase there are several special conditions that must be considered where the basic algorithm fails. If a communication hole exists along the boundary of the event, there may be nodes that are situated along the event boundary yet are not in communication range with a non-event node. In these cases, seeing itself surrounded by nodes which have seen the event the agent will not find a referencing non-event neighbor node to base its decision on the next event node. However, if the agent assumes that it has encountered a hole in the sensor network then it can follow the boundary of that hole by choosing the next node in clock-wise direction. The hole along the boundary may be thus circumvented until the agent reaches a node at the end of the hole with a non-event boundary node neighbor.

Another special condition that may arise during the event boundary discovery phase is the presence of event nodes chains. A chain would consist of a series of event nodes that would only have two or fewer neighboring event nodes. As the agent moves along this chain it would come to a point where non-event boundary nodes could exist on either side of the chain, leading to a ping-pong effect where the agent would bounce back and forth along two points in the chain. In order to avoid this, the agent must detect these chains by recording previously visited nodes and avoid considering them when choosing $node_{n+1}$.

The agent may also get trapped when considering nodes that exist within a concave boundary. In this situation an interior node (**$node_i$**) can falsely be considered as the $node_{n+1}$. To understand these situations consider an agent at **$node_{n-2}$** analyzing its neighbors (including $node_i$ and **$node_{n-1}$**) to determine the next node in the event boundary path. It will sort all neighbors in order of angle and will choose $node_{n-1}$ and ignore **$node_i$** since **$[\theta(node_{n-3} \rightarrow node_{n-2} \rightarrow node_{n-1}))] < [\theta(node_{n-3} \rightarrow node_{n-2} \rightarrow node_i)]$**. However, once the agent moves to **$node_{n-1}$** and then polls all of its neighbors it will find that **$\theta(node_{n-2} \rightarrow node_{n-1} \rightarrow node_i)) < \theta(node_{n-2} \rightarrow node_{n-1} \rightarrow node_n)$** and will therefore choose **$node_i$** as the next node in the agent path. In order to detect this condition all nodes that lie within the region shown within figure 1 must not be considered when determining the next node

(node$_{n+1}$). If the boundary is a concave boundary then these nodes will be those which are interior nodes, and if the boundary is convex then these nodes will be non-event nodes.

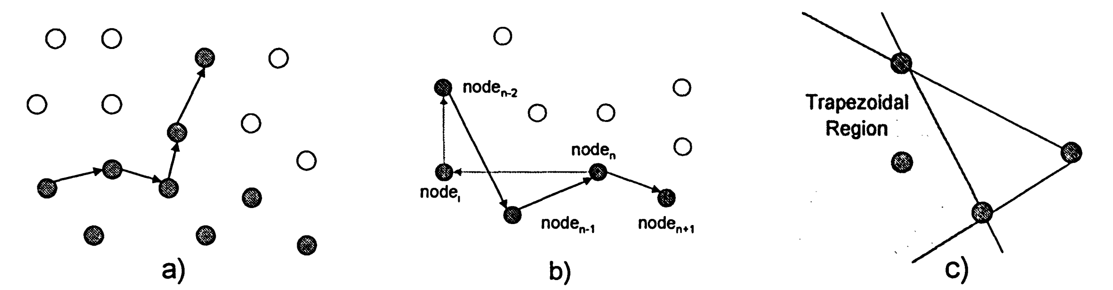

Figure 2: Special conditions for next node selection in event boundary discover. Chain of event nodes (a) concave boundary (b) trapezoidal region for concave boundary node exclusion (c). Note: Event boundary nodes shown in dark gray while non-event boundary nodes shown in light gray while the error path are shown in gray.

Chapter 4: Simulations/Observations

We used Matlab in order to simulate and validate the performance of this algorithm. Since Matlab cannot simulate multiple simultaneous agents processing in parallel, we divided the simulation into distinct time slices. Each time slice represented the time it would take for a node to transmit to all neighbors, receive a response and transmit the agent program code to the next node. The simulation involved defining all nodes in a circular area as having seen an event. Additionally, the network size, density, transmission radius as well as wakeup threshold were all variables that could be varied in the simulation.

The algorithm was tested for variation in efficiency and accuracy against event radius size. Other variables, which have an impact on algorithm efficiency and accuracy, are node density and the wakeup threshold. The node density is the average number of nodes within a square unit of area, and this was fixed at 5 nodes/unit^2. The wakeup threshold was fixed at 0.005 and is the chance that any given event node will generate an agent at the start of a time slice. The algorithm was simulated within a sensor network located within a square area (LxL) with variation in area size from 10x10 to 50x50 units (i.e. L varied in steps of 2 from 10 to 50). The event radius was fixed in relation to the network area with event radius of L/3 (i.e. 3.3 to 17.33 units). For each step in event radius size, we performed three simulations for each step and the average of the results was reported.

An example of the simulation output can be seen in figures 3a and 3b. Within figure 3a the randomized sensor network is shown where all event nodes are marked in black gray while non-event nodes are marked in light gray. Additionally it is shown that the boundary of the event area was (shown in black) was successfully detected. Within figure 3b the agent paths are shown in black (agent 2) and gray (agent 4). Both agents 1 and 3 in this example were terminated before reaching phase 2 of the algorithm since they already encountered another agent.

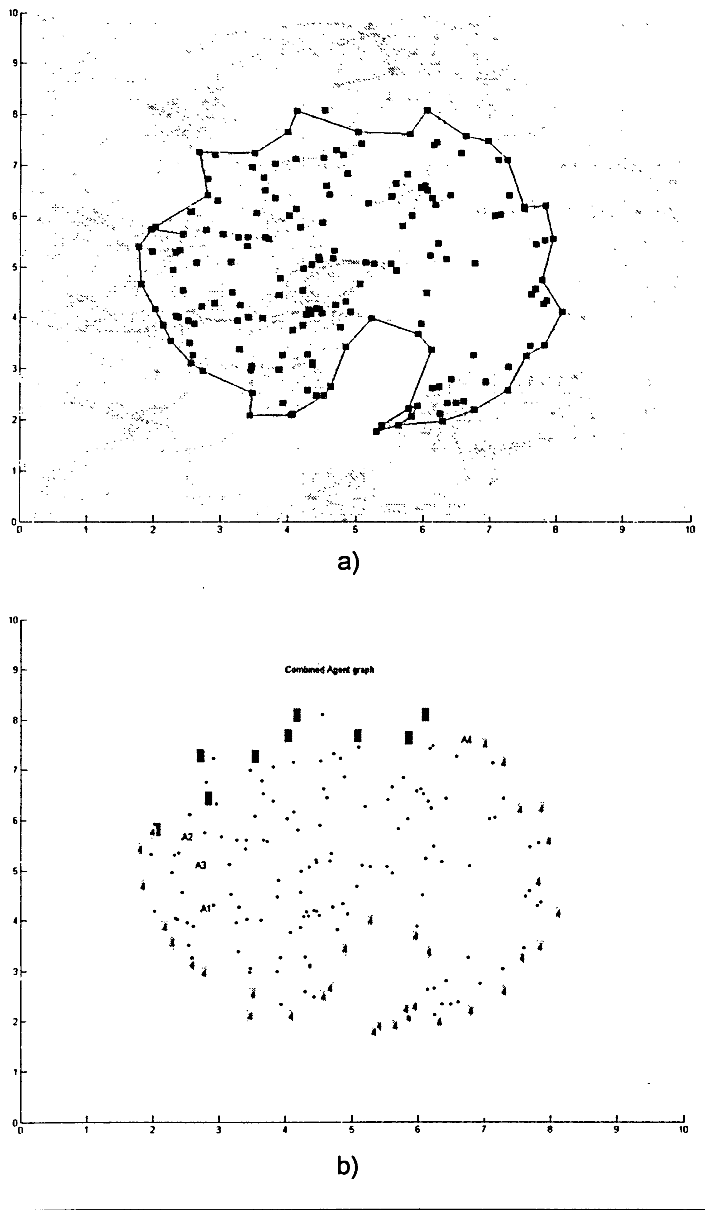

Figure 3: graphical results of simulation output. Event boundary (a) and agent paths (b). light gray lines and nodes depict sensor network and communication paths, black nodes depict event nodes, black line depicts detected event boundary. Note, in figure b) agent starting regions marked as A#, Agent 1 and 3 aborted, while Agents 2 and 4 successfully found the boundary and mapped it.

Event accuracy was measured by calculating the total number of event nodes correctly bounded within the reported event boundary versus all nodes which detected the event. A second statistic, false positives, was used to ensure that no node was incorrectly found as an event node. This was important since this would not be reflected in the accuracy statistic, and could in fact alter the accuracy statistic if present. The event efficiency was broken down into two statistics: efficiency of boundary detection and efficiency of event area detection. The efficiency of boundary detection was determined as the average number of transmissions made per boundary node (i.e. nodes within transmission radius of the boundary). The efficiency of event area detection was determined as the average number of transmissions made by each individual interior or boundary node (i.e. all nodes within event radius + transmission radius).

As shown in the theory section of this paper we theorized that the ratio of boundary nodes to event nodes would decrease as the event radius increased. This concept was demonstrated within figure 4b. Additionally, it can be seen in figure 4a that the relationship of event boundary nodes versus event radius is a linear one while the relationship of event area nodes versus event radius is an exponential one. It should be noted that during simulation the starting algorithm was insensitive to the starting position of the agents.

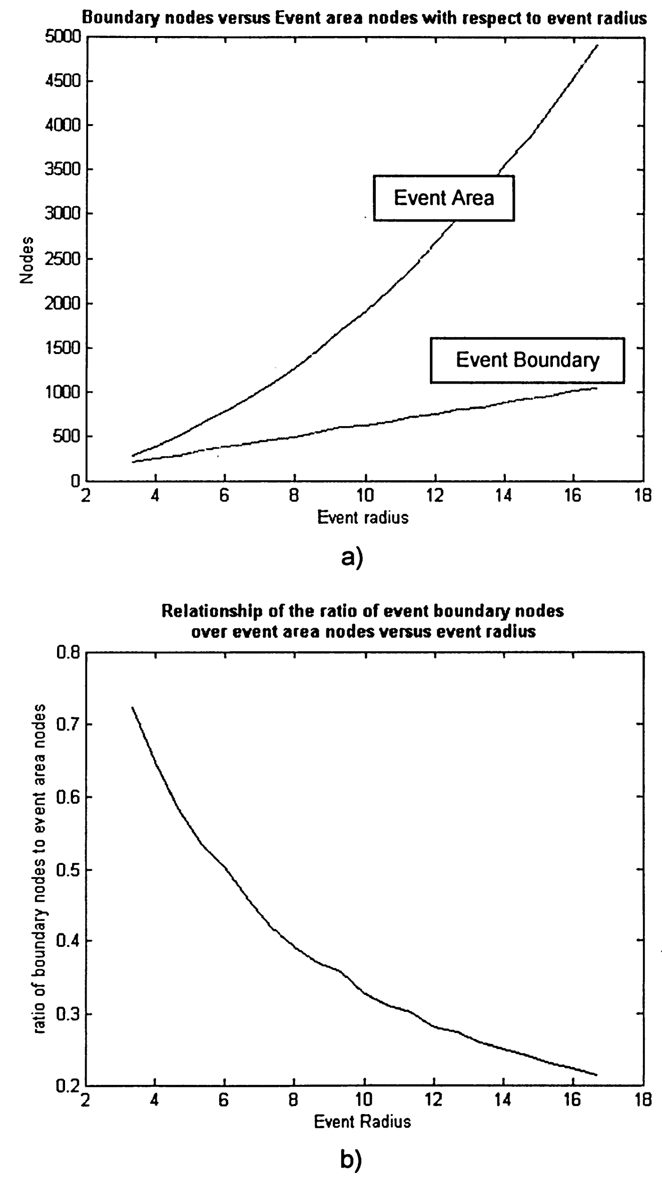

Figure 4: Comparison of the relationship of boundary nodes to event nodes as event radius is increased (a). Relationship of event transmission efficiency to event radius (b)

The accuracy and false positives were found to be 99% and 0% respectively, regardless of the event area size. However, the accuracy and false positives were highly susceptible to smaller node densities, since smaller densities could cause small isolated pockets of event nodes along the boundary. This condition was not further examined since, for this paper, it was assumed that the event area would be homogenous where all event nodes are interconnected.

Furthermore the efficiency of the algorithm was theorized to increase as an inverse function of the event radius, similar to the ratio of boundary nodes over area nodes as shown in figure 4b. Additionally, it can be observed that the efficiency of boundary node transmissions was constant as shown in figure 5, demonstrating that the majority of the algorithm's functions were seen in the event boundary.

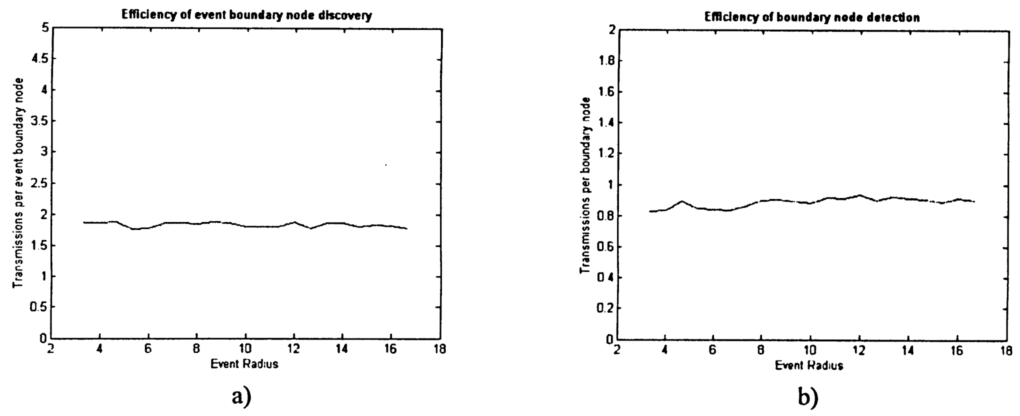

Figure 5: efficiency of event boundary node transmissions (a) and all boundary node transmissions (b).

Finally, the simulations do not take into account the overall reduction in overhead required to report the boundary back to the sink. Considering that the overall size of the sensor network can not be pre-judged as well as the transport algorithm, it becomes impossible to predict the number of transmissions reduced when reporting to the sink. However, that being said it is possible to conclude that the overall number of nodes requiring transmission has been significantly reduced, from all nodes that have seen the event boundary to only a select few nodes that are the final resting locations for the agents.

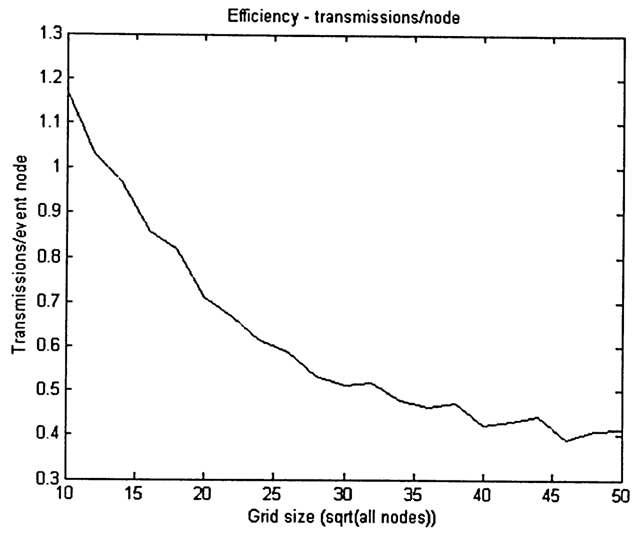

Figure 6: Transmissions/event node as measured by the simulation for varying event radiuses

Chapter 5: Conclusions

We can conclude from the simulation results that the algorithm operated as expected with the assumptions and restrictions given. Additionally, we have shown that a homogenous event boundary can be successfully mapped using the proposed algorithm with 99% efficiency. Furthermore, with the use of agents to facilitate boundary discovery, the majority inter-node transmissions were successfully restricted to the boundary region for network of sufficient size. The main value of this algorithm is that transmission efficiency is not exponentially related to event radius size but instead linearly related to it, allowing for the algorithm to be much more scalar then traditional algorithms.

It is important to note the effects that the assumptions have made on the results. The density of the sensor network, highly affects the contiguity of the event area. If the event area is not a contiguous region, it has the risk of being recognized as multiple events or have some of its regions unrecognized at all. However it should be noted that this could be solved if 2nd hop neighbor event polling is used with the drawback of impacting efficiency.

Additionally, the effect of density and the wakeup threshold has a large effect on the number of agents generated. If we assume that the density of nodes is sufficient that the area remains contiguous, then we can see that as the event size increases the number of nodes that "wake up" and generate agents will increase. However, if we look at the reverse case, where the event size is decreasing, then there will come a point where the network will not generate any agents. Therefore for any threshold value given (as well as density), there will be a minimum event area resolution, whereby any event smaller then this resolution may not be detected at all. This can in some cases be considered an advantage, since in many situations a minimum event size threshold may be acceptable. Also, if the event exists within the network for a prolonged period of time, other time-slices may see the event.

Another limitation of this algorithm is that it does not estimate the exact boundary of the event, but instead resolves the minimum detectable boundary presented by the network. This limitation can be addressed if the algorithm is adapted to include some filtering, and boundary estimation.

Finally, in this paper we do not take into account the effects of failed nodes or nodes which incorrectly determine that they have viewed the event. It can be seen however, that future work could occur where the algorithm could utilize 2nd hop neighbor information as well as filters such as those used in [4] to ignore false positives/negatives.

References

[1] K.K. Chintalapudi and R. Govindan, "Localized Edge Detection in Sensor Fields", in IEEE Ad Hoc Networks Journal, pp. 59-70, 2003. 2]

[2] R. Nowak, U. Mitra, "Boundary Estimation in Sensor Networks: Theory and Methods" IPSN 2003: 80-95

[3] S. Li, S. H. Son, and J. A. Stankovic, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks", in IPSN 2003, Palo Alto, USA, April 2003.

[4] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks", in IEEE Transactions on Computers, Vol. 53, No. 3, pp. 241-250, March 2004.

[5] Pei-Kai Liao, Min-Kuan Chang and C.-C. Jay Kuo, "A distributed approach to contour line extraction using sensor networks," in Proc. IEEE VTC Fall 2005.

[6] Thomas Clouqueur, Parameswaran Ramanathan, Kewal K. Saluja and Kuang-Ching Wang, "Value-Fusion versus Decision-Fusion for Fault-tolerance in Collaborative Target Detection in Sensor Networks", in Proceedings of the 4th Ann. Conf. on Information Fusion, pp. TuC2/25-TuC2/30, Aug 2001

[7] A V U Phanikumar, Adi Mallikarjuna Reddy V, D.Janakiram and S. Priya, "Distributed Collaboration for Event Detection in Wireless Sensor Networks", in Proceedings of 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC05) Dec, 2005

[8] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized Fault-Tolerant Event Boundary Detection in Sensor Networks", IEEE INFOCOM, March 2005

[9] Daniel J. Abadi, Samuel Madden, Wolfgang Lindner, "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks", in Proceedings of the 31st

International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, September 2005

[10] Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty, "Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents", in Proceedings of 5th International Conference on Information Fusion, Annapolis, MD, 2001.

[11] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), August 2000.

[12] J. Kulik , W. Heinzelman , and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," Wireless Networks, March 2002.

[13] A. Manjeshwar and D. P. Agarwal, "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks," In 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2001.

[14] A. Manjeshwar and D. P. Agarwal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, pp. 195-202.