UCLA
Papers

Title
Adaptive Sampling for Marine Microorganism Monitoring

Permalink
https://escholarship.org/uc/item/2x64v9w9

Authors
Zhang, Bin
Sukhatme, Gaurav
Requicha, Aristides A. R.

Publication Date
2004-05-05

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Adaptive Sampling for Marine Microorganism Monitoring

Bin Zhang, Gaurav S. Sukhatme and Aristides A. G. Requicha
Computer Science Department
University of Southern California
Los Angeles, CA 90089
Email: binzhang@cs.usc.edu, gaurav@usc.edu, requicha@usc.edu

***Abstract*—We describe the design and construction of an underwater sensor actuator network to detect extreme temperature gradients. We are motivated by the fact that regions of sharp temperature change (thermoclines) are a breeding ground for certain marine microorganisms. We present a distributed algorithm using local communication based on binary search to find a thermocline by using a mobile sensor network. Simulations and experiments using a mote test bed demonstrate the validity of this approach. We also discuss the improvement in energy efficiency using a submarine robot as a data mule. Comparisons between experimental data with and without the data mule show that there are considerable energy savings in the sensor network due to the data mule.**

I. INTRODUCTION

With advances in processor and radio technologies, low-price wireless sensor and actuator networks are becoming available. With this new technology, a large number of low cost sensors and actuators can be deployed to provide focused in-situ sensing. The integration of local processing and storage allows nodes within such a network to not only provide raw data but also draw inferences and provide high level information. Wireless sensor networks have many applications, such as habitat monitoring [2] [10], battle field target tracking, in-situ exploration of gaseous biosignatures [4] and chemical plume tracking[8]. In this paper, we focus on a particular application: Marine Microorganism Monitoring, which we introduce next.

Microorganisms such as Phytoplankton are exceedingly small (2-3μm) and are distributed in the ocean at varying spatial scales. It is not practical to locate them by measuring their density everywhere. Both from the point of view of locating marine microorganisms, and from the point of view of studying their behavior, it is beneficial to study how their numbers and location are correlated with chemical (e.g. nutrient concentration) and physical parameters (e.g. temperature, light intensity) in the marine environment. There are two major factors that are important to the growth of microorganisms: light intensity and nutrients. In the ocean, the former comes from above (sunlight) and the latter comes from below. At a certain depth, there is a good balance between light intensity and nutrients, and the density of certain microorganisms may be expected to be high. In the ocean, such a region could be a thermocline, a zone where seawater temperature drops rapidly. This sharp change in temperature acts as natural barrier to nutrient diffusion.

Given the hypothesis that marine microorganisms bloom at a thermocline (a physically measurable phenomenon), we focus on the detection and localization of a thermocline in an underwater environment. We propose a decentralized approach - distributed binary search - to localize a thermocline using a wireless sensor actuator network. The spatial gradient of temperature induces a scalar field over all locations underwater. Formally a thermocline is a level set of this field (a locus of points in the environment) with the property that no other level set has a greater value. In practice we look for a family of level sets whose field values exceed some pre-specified threshold.

This paper presents two key achievements:

1) We describe an **adaptive sampling algorithm based on binary search** for the network to reliably detect a thermocline. The algorithm is based purely on local communication, and has been implemented and tested in experiments underwater.
2) We experimentally establish and characterize the network energy savings due to the **usage of a robot submarine as a data mule** for the sensor network.

II. THE ADAPTIVE SAMPLING ALGORITHM: DISTRIBUTED BINARY SEARCH

Our goal is to develop an algorithm for sensor networks working underwater where communication range is very limited. Given the immense size of the ocean and the tiny size of microorganisms, it is not practical to achieve high sampling density by increasing the density of sensors. By allowing the nodes to move, we can achieve high resolution with significantly lower sensor density.

The basic idea of our approach is sampling by divide and conquer. In this paper we address the problem in one dimension, namely depth. Suppose we have n nodes deployed in a vertical array where the topmost node is connected to the external world (is an 'edge node'). Each node has its own processor, memory, temperature sensor and radio. However, the communication range is limited and each node can only communicate with its nearby nodes. Each node also has a pressure sensor. Since the change of water pressure is linear in the change of depth, by measuring the pressure around it, the node is able to estimate its depth. We also assume that nodes are able to change their depth. This can be achieved by change the buoyancy of the node.

The search space is 1D, and is divided into regions. Every node uses its ability to move to explore one such regions. The process is refined by splitting regions into halves i.e. binary search. Each node communicates with its neighbors and tries to persuade them that the thermocline lies within its search region. A process of data aggregation is enacted on the route from each node to the user to combine the conclusions (about the thermocline location) arrived at by the various nodes.

A. Binary Search

Binary search is exploited to find local temperature gradient maxima. At initialization, each node n_i collects temperature data at both end points of its search space, i.e., the upper-most point and lower-most point. The temperature and the depth at each point, t_t, t_b, p_t and p_b, are noted. Then, the node changes its depth and moves to $\frac{p_t+p_b}{2}$, where it collects a new temperature reading t and depth p. The new point divides the search space of node n_i into two parts, the upper part and the lower part. The differences between the new reading and the two previous readings are calculated.

$$\Delta t_t = |t_t - t| \tag{1}$$

and

$$\Delta t_b = |t_b - t|. \tag{2}$$

If $\Delta t_t > \Delta t_b$, the lower part is discarded, and t_b, p_b are replaced by t, p. Otherwise, the upper part is discarded, and t_t, p_t are replaced. The remaining part of the search region is the new search region. This process is repeated until termination conditions are satisfied (based on sensing resolution).

B. Data Aggregation

Data Aggregation [9] is an important approach to improve the energy efficiency for sensor networks, which is critical if the sensor networks need to operate continuously for a long time without human attendance. Instead of sending raw data directly to users, some data processing is done 'in network', and only the processed data is sent back. Normally, the size of the latter is much less than that of the former. We define four messages for data aggregation:

```
BUILD-ROUTING-TREE
REGISTRATION
QUERY-MAX-GRADIENT
GRADIENT-REPORT
```

Messages BUILD-ROUTING-TREE and REGISTRATION are used to temporarily build a tree expanding all the nodes in the network. BUILD-ROUTING-TREE is used to initialize the process while REGISTRATION is used to build the tree. QUERY-MAX-GRADIENT is the original query from the user and is used to query nodes on maximum gradient. The last message is the most important, and it is the basis of data aggregation. The format of GRADIENT-REPORT is as follows:

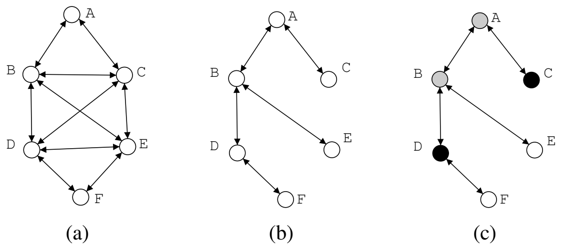
```
typedef struct {
  short pos;
  short tempDiff;
  short id;
  short posDiff;
  short temp;
} Gradient_t;

typedef struct {
  char num;
  Gradient_t gradients[MAX_NUM];
  short discardThreshold;
  short discardAreaSize;
  short crc;
} ReportGradient;
```

This message indicates the maximum gradient (`tempDiff`), its location (`pos`), the node who found it (`id`) and current resolution (`posDiff`). `MAX_NUM` defines the maximum number of max gradient one message can report.

In our approach, before data aggregation starts, a routing tree needs to be built. On receiving an initialization message from a user, one node, such as node A in fig1(a), sends the message BUILD-ROUTING-TREE with its own ID and the boundaries of its search space to its neighbors to initialize the construction of the routing tree. We assume that each node already knows its search space and the reliable communication range under water before run time. Any node receiving that message, for example node B, would first check the maximum distance between A and B. If the maximum distance is less than the reliable communication range and B does not have a parent, it sends the message REGISTRATION to node A. If the message successfully reaches node A, node B sets A as its parent. Otherwise, node B will wait for another BUILD-ROUTING-TREE message. On receiving the message REGISTRATION, node A would put node B in its child list. Then node B forwards the message BUILD-ROUTING-TREE to its own neighbors and waits for REGISTRATION messages. If a node does not receive any registration, it is a leaf. Finally, a tree is built and the node which received the query from users would be the root, as shown in fig1(b).

Though nodes are able to move, each of them moves just within a small area, i.e. individual search space. The edge of the routing tree only exists between two nodes between which the maximum distance is less than the reliable communication range. So, the movement of the nodes would not affect routing tree, and hence the routing tree would be valid throughout the execution of the algorithm.

Besides the parent id and child list, each node also keeps three other variables: `selectedChildList`, Δt_d, and Δp_d. `selectedChildList` keeps the list of active children. Together with Δp_d, Δt_d define the maximum temperature gradient discarded in the past. Δt_d is initial-



(a) (b) (c)

Fig. 1. Different stage of Distributed Binary Search 1(a) The topology of the sensor network, which demonstrates which nodes can communicate with each other 1(b) The topology of the sensor network after the routing tree has been built. The nodes without connection now would ignore the messages from each other even if they actually can receive the message from the other 1(c) After several steps, most nodes would be inactive, and only the node within who may find the thermocline would remain active.

ized to 0.

When a node in the network (say B) receives a query message QUERY-MAX-GRADIENT, it forwards the message to its children. At the same time, it starts the local maximum gradient search. However, it just executes one step, and then waits for responses from its children. After it receives replies from its children, node B compares the conclusions of its children and its own. The reply from children has the format of MAX-GRADIENT-REPORT. As mentioned above, the message has the field `discardThreshold` and `discardAreaSize`, which together provide the estimation of the greatest temperature gradient discarded by this child or its children. Node B first updates its own fields Δp_d and Δt_d. If

$$\frac{\Delta t_d}{\Delta p_d} < \frac{discardThreshold}{discardAreaSize} \tag{3}$$

it replaces Δt_d and Δp_d with `discardThreshold` and `discardAreaSize`. Then, node B updates its own format based on the report from its children and its own observation. The candidate thermoclines would be examined, including the one calculated by node B itself. Any one whose gradient is less than $\frac{\Delta t_d}{\Delta p_d}$ would be discarded. If the remained thermoclines are more than the report can accommodate, those with the greater temperature gradient will be kept. Then the report is sent to node B's parent.

The reason to keep track of the maximum discarded temperature gradient is to suppress unnecessary local maximum search and messaging. Suppose node B discarded a subregion S_B, where the estimated temperature gradient is Δp_B. When node B discarded the subregion S_B, it is assumed that the area with estimated gradient Δp_B can not be the area with global maximum gradient. So, any other area with estimated gradient less than that can not have the global maximum gradient, either, and it is safe to drop those areas. In this way, many nodes will be suppressed and become inactive. That is, they would stop local maximum searching and negotiation. If they do not have to forward messages from root to the active nodes, the inactive nodes can go to sleep, saving energy.

After sending out the message GRADIENT-REPORT to its parents, node B updates the variable `selectedChildList`. If the data from a child of B (say D) is actually forwarded by node B as part of its report, node D would be on the list of selected children. When a new query comes, node B would forward the query to the selected children.

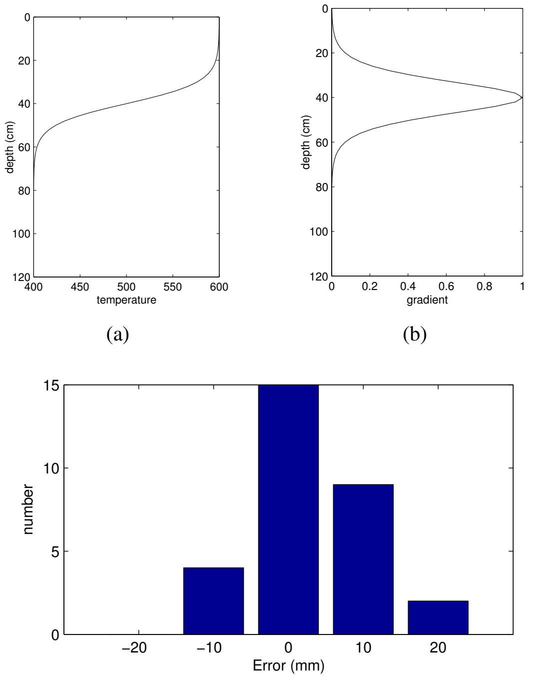
C. Overall Approach

The whole mechanism for thermocline detection is as follows. When a user needs to locate the thermocline, he sends an initialization command to any one of the nodes. The node which receives the command broadcasts the message BUILD-ROUTING-TREE, and begins to build the routing tree. Finally, a tree is built with that node as root. Then, every node will explore the upper-most point and the lower-most point of its search regions to initialize t_t, p_t, t_b and p_b. At this point the user can send the query to find the thermocline. The root of the routing tree will forward the message. Any node which receives this query start a local binary searching and also forwards the message to its children. On finishing one step of local search, each node waits for its children to reply to the query. By combining the reply from children and its own data, each node sends its report to its parent. The node would also keep track of the selected child list. Finally, there is one report from the root to the user. This process is repeated, every successive report has a better resolution on the thermocline location than the previous one. When the preset resolution is achieved, or there is no improvement of the resolution, the algorithm stops.

D. Improvement with a Data Mule

After the first one or two steps of the distributed binary search, most nodes become inactive. However, they have to be awake if they are on the path from the active nodes to the root of the routing tree since they are needed to forward messages from the active node to the users. When binary search is running, the nodes in the network can be divided into three groups, as shown in 1(c). The nodes labeled black are active nodes while the white ones are inactive nodes and may go to sleep. The gray nodes are the nodes which are not active but lie on the path from an active node to the root. Those nodes must keep awake to forward messages. If we can create a short cut from the active nodes to the root, all other nodes can go to sleep, and energy can be saved.

Because radio signals attenuate rapidly underwater, long range communications may not be achieved by using radio. One way to solve this problem is to use sound instead radio for communication. Another way is to use a messenger, a robotic node that can move itself autonomously. This robot would move from the neighborhood of the root to the active node, and forward the messages from one to another, thus acting as a data mule. Motion consumes energy; but we assume that a process exists to recharge the robot when it surfaces, and do not analyze it further here. Given such a process we ask if the introduction of a data mule reduces energy consumption of the static network.



(a) (b)

(c)

Fig. 2. 2(a) simulated temperature profile 2(b) simulated temperature gradient alone the depth and 2(c) the summary of 30 simulations with different thermoclines. Y axis indicates the number of simulations, and the X axis indicates the errors.

III. SIMULATIONS

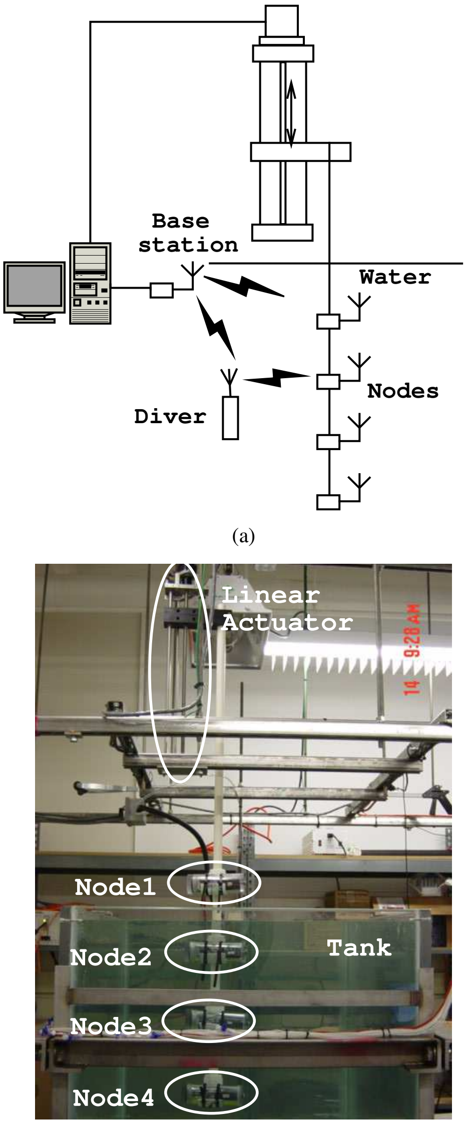
In this section, we discuss the simulations of the distributed binary search. In our simulation, 4 nodes were deployed alone one vertical line. Each node has the abilities mentioned in section 2, such as communication, limited mobility. The reliable communication range is set to be 70 cm, and the width of the search space of each node is 30 cm. The temperature profile was simulated by the following formula

$$T = \frac{k_T}{1 + \exp(k_Z \cdot (Z - Z_0))} \quad (4)$$

where T is temperature, Z is depth, Z_0 is the center of the thermocline and k_T and k_Z are scaling parameters. In our simulation, k_T and k_Z are 200 and 0.02 respectively. Fig 2(a) demonstrate the typical temperature profile and fig 2(b) demonstrates the temperature gradient versus depth.

At the beginning of each simulation, the 4 nodes are at depth 0, 30, 60, and 90 cm and waiting for commands. A client program start the algorithm by sending the query messages to one of the 4 nodes. the simulation was repeated 30 times and each time we chose a different thermocline. To be more specific, the center of the thermocline, Z_0, changed from 20 cm to 50 cm. We collect the estimated depth of the thermocline and compare it with the actual one, Z_0. Fig 2(c) demonstrated the distribution of the estimation errors.

From fig 2(c), we see that the errors of 50% estimations were less than 5mm, and those of 90% estimations were less than 15mm. Given the k_T, k_Z above, the maximum gradient alone the depth is 1. When $|Z - Z_0| < 15$, the gradient is greater than 0.9778. That is, the gradient difference is less than 2.22%. So, the error of ±15 is acceptable.



(a)

(b)

Fig. 3. (a) A schematic of the test bed and (b) The experimental test bed

IV. EXPERIMENTS

A. Experimental Setup

We have built an experimental test bed (fig 3) to validate the distributed binary search algorithm. The test bed consists of 5 Mica2 motes, one PC and one linear actuator with controller.

Motes [6] were designed at UC Berkeley to provide a platform for research on sensor networks. Mica2 is one of the new versions, and it consists of an 8-bit atmel ATMega128L microcontroller, 128k flash memory, CC1000 radio working at the frequency of 433MHz. Each mote has seven 10-bit multiplex ADC channels. By attaching different sensor boards, one mote is able to access different sensors, such as light sensors, thermistors or accelerators. Our test bed uses the basic sensorboard, which consists of one light sensor and one thermistor.

The configuration of the test bed is shown in fig 3. Four of the five motes are attached to a rigid tether, which is, in turn, attached to the linear actuator. Each mote can communicate with its neighbors over the radio, and hence they compose a simple wireless sensor network. However, no mote can talk to all other motes underwater since the communication range of the radio reduces greatly underwater. For example, the top mote can not send a message to the bottom one directly. The fifth mote is the base station, which serves as the bridge between the PC and the sensor network. The PC is an interface between users and the network. During the experiments, a client program running on PC would start the algorithm and refine the estimation of the thermocline by sending QUERY-MAX-GRADIENT to the sensor network.

In section 2, we assumed that each node of the sensor network has a pressure sensor to measure the depth. We also assumed that each node has limited mobility. However, in our test bed, none of those node has pressure sensor, and it is obvious that all the nodes in the network share one degree of freedom. To implement the distributed binary search on our test bed, following methods are taken so that our test bed can simulate the system mentioned previously. When a node needs to move to a certain depth, it sends the PC a message, Motion-Command, which indicates the destination of the node. If this node is the only active node, on receiving the message, PC would control the linear actuator and move the node to that depth. After the movement is done, a message would be sent from the PC to the node to indicate that its requirement was fulfilled. We name this message as Motion-Done. However, there may be two or more active nodes. In this case, the Motion-Command messages are put in a queue. The PC picks one command from the queue each time, executes it and then send Motion-Done message to the node which sent the command.

The message Motion-Done contains information on current position of linear actuator. Since all the nodes are attached to the linear actuator, given the position of one node on the linear actuator and the position of the linear actuator, it is easy to compute the depth of the node. This is the way how each node measure its depth.

To save energy, message Motion-Done is actually combined with the message QUERY-MAX-GRADIENT, and the message Motion-Command is combined with the message GRADIENT-REPORT. In summary, the algorithm implemented on the test bed goes in the following way: After the routing tree is built, the client program send the QUERY-MAX-GRADIENT, which contains the initial position of the linear actuator, to the sensor network through the root of the routing tree. On receiving this message, each node calculates its current depth. Then, every node reads its thermistor and compute the gradient. After the interaction among the nodes, the reports from some nodes would arrive the PC. the client program would extract current estimation of thermocline from those messages. It also collects the Motion-Commands of those nodes and put them in the queue. After the PC executes the first Motion-Command, it would send the another QUERY-MAX-GRADIENT message to corresponding node. On receiving this message, the node calculates its current depth, and proceed one more step of the algorithm. When PC receive the message GRADIENT-REPORT from this node, it append the extracted destination to the rear of the queue, and execute the Motion-Command at the front of the queue.

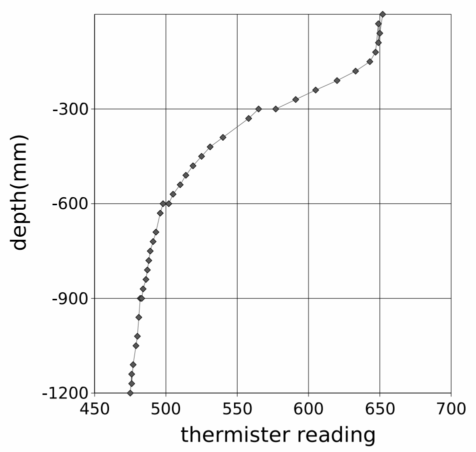


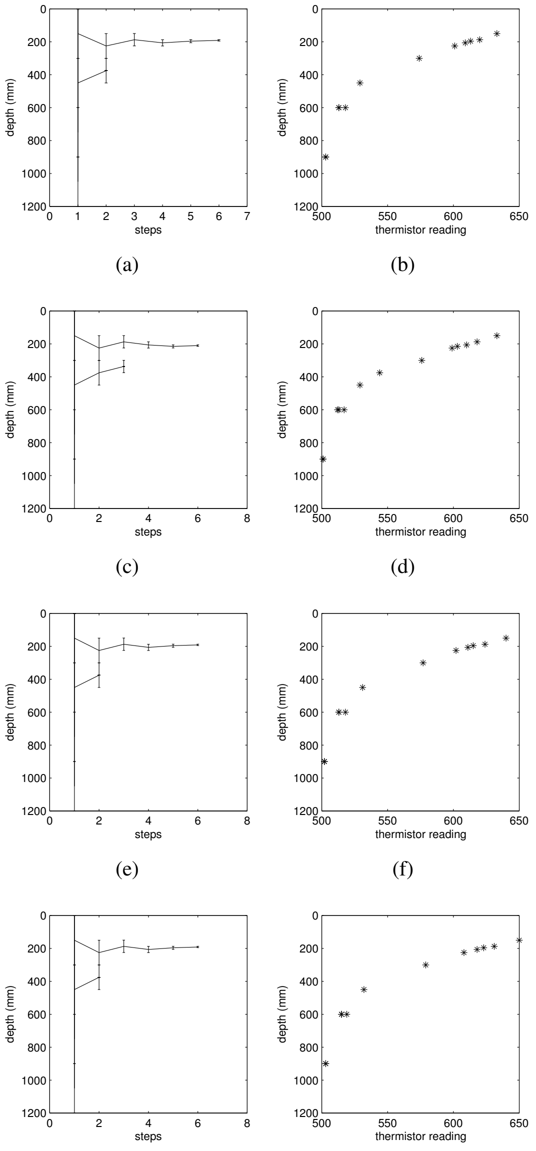
Fig. 4. Temperature profile drawn with the data collected by taking the temperature readings every 30 mm

Our algorithm is implemented atop TinyOS [6] and SMAC [13]. TinyOS is the operating system developed for motes; it provides a task scheduler as well as an API to the hardware, such as radio and ADC. SMAC is an implementation of radio stack [13]. With SMAC, the radio is put to sleep automatically, and the size of the message can be varied. All the experiments were carried in a tank filled with water. A thermocline was created by a heater in the tank. The heater was put in the water just beneath the surface so that water temperature is not constant along depth. As shown in fig 4, at depths between 200mm and 400mm, the temperature dropped rapidly, which is a thermocline.

B. Experimental Results

First, we conducted a series of experiments to test whether the distributed binary search is able to reliably localize the thermocline in the tank. 24 experiments were carried, and fig 5 shows the results of 4 of those experiments.

In each picture, the experimental results are shown as four curves with error bars. Each curve corresponds to one node. At each step of the binary search, every active node reports back where it believes the thermocline is, and the width of the thermocline. Each point on the curve is a candidate location of a thermocline and the associated error bar is the reported width. Inactive nodes do not report, and



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Fig. 5. Results of 4 experiments. The figures on the left are the output of the distributed binary search, and the figures on the right demonstrate the corresponding temperature profile.

hence the corresponding curves terminate. At the beginning of the process, every node is active, and all of them report back. In addition, all nodes initially report the width of thermocline as that of its search space. After the first step, only 2 nodes are active. That is why there are 4 curves at the beginning, but only 2 curves left after the first step. Those 2 nodes would continue competing with each other until one of them becomes inactive or resolution reaches the preset threshold. The former case means that all the nodes in the network agreed that the thermocline is within the search space of only the active node. For example, in all of the four experimental trials shown, all the nodes finally agreed that the thermocline is within the search space of node 1. Depending on the temperature profile, the steps needed by the node to reach that agreement may vary. For example, it took the nodes 3 steps in the experiment carried out at 5(c), while it only took 2 steps in other experiments.

C. Improvement with a Robotic Data Mule

As discussed in section 2, a data mule can create a short cut from the base station to the active nodes. A series of experiments were conducted to measure the energy saved by using a data mule.

A robotic mote-based submarine (6) was developed in our lab [1], and it was used in our test bed as a data mule. The submarine is composed of a plastic container, a linear actuator controlling a cylinder, a pressure sensor and a Mica2 mote. The mote measures the pressure reading and controls the linear actuator. By changing position of the piston in the cylinder, the mote can change the buoyancy and hence can move the submarine up or down. The submarine is capable of depth regulation.

In sensor networks, the majority of the energy is actually consumed by the radio. According to the data sheet of Mica2, the current draw for full mode cpu is 8 mA while that for sending message is 27 mA. Therefore, the number of messages passed between nodes indicates the energy consumption, and hence indicates the life of the sensor network. We counted the number of messages sent and received by each nodes in two scenarios. In one scenario, the settings of the experiments are the same as the experiments described in the previous section. In another scenario, the data mule was used to reduce the number of messages exchanged.

Table 1 shows the experiments results. The experiment with the data mule was repeated 3 times and the numbers shown in the table are averages. The last row of the table shows the number of messages broadcasted, which are not counted as the received messages of individual nodes.

From the data we collected, it is obvious that the data mule reduced the number of messages exchanged between nodes, and hence saved the energy consumed. Most of the messages reduced belong to node 3 or node 4. In our experiments, node 4 is the node that is closest to the base station, and it is able to communicate with the base station directly. Node 2 and node 3 can communicate with node 4 directly but node 1 need node 3 to forward the messages to node 4. The thermocline is within the search space of node 1, and node 1 is the active node throughout the whole experiment. When there is no data mule, all the messages from and to node 1 must go though node 3 and node 4. So, in the first scenario, node 3 and node 4 sent and received lots of messages. When the data mule is used, a short cut was created between the base station and node 1. In this case, node 3 and node 4 did not send or receive any messages after the first step, when they became inactive. Finally, the messages sent and received by them reduced to almost half compared to the previous scenario.

V. DISCUSSION

A. Interpreting the Experimental Results

In the experiments described above, the outputs of the distributed binary search are all from one node. However, this may not always happen. In certain cases, more than one

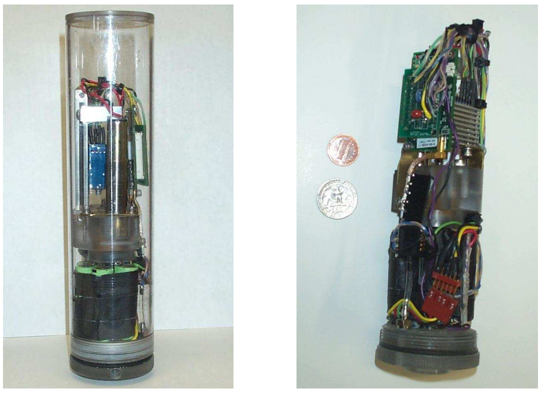

(a) (b)

Fig. 6. The robotic mote-based submarine

TABLE I
NUMBER OF MESSAGES EXCHANGED

	No Data Mule Used		One Data Mule Used	
	N_{sent}	$N_{received}$	N_{sent}	$N_{received}$
Node1	9	8	9.33	8
Node2	4	8	8	11.67
Node3	14	23	3.67	8.33
Node4	28	26	18	16.67
Data Mule	NA	NA	8	11.33
Base Station	10	12	9	11.33
All	N/A	12	N/A	12.33

node may report even if the resolution limit is reached, or the difference between estimated gradient of the two sub-search space is so small that the nodes can not distinguish them with their 10-bit ADC. For example, in the simulation where $Z_0 = 300$, both GRADIENT-REPORT messages from node 1 and node 2 keep reaching basestation till the end of the algorithm.

There are two interpretations for the case when more than one node reports at the end of the algorithm. The first interpretation is that there are more than one thermocline and their gradient values are almost the same. In that case, the nodes which reported are likely located far from each other, and the reported areas are not adjacent to each other. The other interpretation is that the thermocline is at the border to adjacent search areas. The simulation mentioned above is one example. In that case, those areas are adjacent to each other. With a clustering algorithm, this interpretation can be done automatically.

B. Analysis of the Number of Message Exchanged

If we assume that all the nodes in the network are attached to one rod, and each node can only exchange messages with its immediate neighbors, then the number of messages exchanged between nodes can be calculated.

During the initialization phase, BUILDING-ROUTING-TREE and INITIALIZATION need to be injected into the network. Those two messages flood the network, and any node receiving them forwards them. Additionally, each node which gets the BUILDING-ROUTING-TREE message needs to send at least one REGISTRATION message. One exception is the node connected directly to base station, and that node does not send the message REGISTRATION. In our test bed, the PC needs to inform each node of its position and hence 2 messages are injected, one for the uppermost point and one for the lowermost point. During the initialization phase, the node need to move itself twice, the top position and bottom position, and 2 messages would be sent from nodes to the PC. So, during the initialization phase, each node, other than the one that communicates to the base station directly, would send $2 + 2 + 2 = 6$ messages.

Next we consider the binary search phase. At each step, one message informing node position is sent from the base station to active nodes and one MAX-GRADIENT-REPORT would be sent from the active nodes to the base station. If there is no data mule, every message is sent multi-hop. Those nodes between the active nodes and base station would receive and send 2 messages at each step. An active node would only receive one and send one message if it is not on the path from other active nodes to the base station. Those inactive nodes not on the path do not have to receive or send any message. So, the nodes which need to forward messages between its children and its parent would send and receive most messages. If we assume that before the end of binary search, m nodes were selected as active nodes, and the i-th active node reported k_i times. The max number of messages sent and received by one node are

$$N_{received} = 6 + \sum_{i}^{m} 2k_i \tag{5}$$

$$N_{sent} = 6 + \sum_{i}^{m} 2k_i \tag{6}$$

If there is a data mule, one short cut would be created between the active nodes and the base station. In this case, there are no nodes except the messenger on the path between the active nodes and the base station, and the nodes that would send and receive most messages are the active nodes. Because those nodes would send and receive one message at each step, the maximum number of messages sent and received by one node are

$$N_{received} = 6 + k \tag{7}$$

$$N_{sent} = 6 + k \tag{8}$$

where $k = max(k_1, k_2, ..., k_m)$. Therefore, the data mule reduce the number of messages exchanged between nodes. The longer the system runs, the more the active nodes, the more messages reduced by the data mule, and the more energy saved. We reiterate that this does not imply a net savings in energy - after all the data mule needs to be recharged from time to time. However we believe this is desirable (easier logistically to charge one robot instead of many nodes).

VI. RELATED WORK

The problem most similar to the one studied here is edge detection, which has been studied for decades in the computer vision [5] community. Several approaches have been developed. However, given the density of the samples in images and without the constraint of energy consumption, gradient of any point can be calculate easily and the resolution is high.

Another similar problem, tracking chemical plumes, has been studied by [8]. A dual-space approach has been proposed to estimate the boundary of the chemical plumes. However, that approach is a centralized algorithm and all the node need to send the raw data to a computer before the algorithm can be executed.

Recently, [3] proposed three approaches for edge detection using a sensor network. The first approach is a statistical approach while the other two are based on a high pass filter and a classifier respectively. In all three approaches, each sensor gathers information from its neighbors, and independently determine whether it is on the edge of an event. One assumption taken by all those approaches is that every node in the network is able to detect a given event just by its own sensor reading. Unfortunately, this assumption is not always true. For example, in the case of thermocline localization, it is impossible to define high temperature and low temperature since they may corresponds to the same thermistor reading in different places.

An approach based on hierarchical structure of "clusterheads" was proposed by [11] to estimate the boundary. With the hierarchical structure, fine resolution would be achieved alone the boundary, while the resolutions in homogeneous regions are coarse. However, with all the sensors static, the resolution of the boundary is still bounded by the density of sensors.

Data aggregation used in this paper is a well known method in sensor networking to reduce energy consumption. The idea of building a routing tree is due to the TinyDB work [9]. However, no history information was kept in TinyDB, and every query is flooded throughout the whole network. This is not an efficient way to locate a thermocline. So, we exploited the idea to keep historical information and only forward the query to the nodes which have the potential to affect the results. This is similar to the reinforcement on the path from source to sink used in Directed Diffusion [7].

VII. CONCLUSION AND FUTURE WORK

We have described an adaptive sampling algorithm based on binary search for the network to reliably detect a thermocline. The algorithm is based purely on local communication, and has been implemented and tested in experiments underwater. We have experimentally established and characterized the network energy savings due to the usage of a robot submarine as a data mule for the sensor network.

A limitation of the present approach is that it assumes a static thermocline. This is not unreasonable, since the thermocline created in the tank moves gradually (over a period of a few hours). However, after the thermocline moves, the whole algorithm needs to be executed again to re-locate it. We are working on a revised implementation of the algorithm, which would track the change of the thermocline with only a few relevant nodes involved. We are also working on extending the algorithm to generalized 3D sensor networks.

ACKNOWLEDGMENT

The authors thank Carl Oberg, Vitaly Bokser and Gaurav Sharma for their help with the test bed and Prof. David Caron and Beth Stauffer for their comments. This work is funded in part by grants CCR-0120778, EIA-0121141, IIS-0133947 from the National Science Foundation.

REFERENCES

[1] V. Bokser, C. Oberg and G. Sukhatme, and A. Requicha. A Small Submarine Robot for Experiments in Underwater Sensor Networks *International Federation of Automatic Control Symposium on Intelligent Autonomous Vehicles*, November, 2004.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton and J. Zhao. 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.

[3] K. Chintalapudi and R. Govindan. Localized Edge Detection in Sensor Fields. In *IEEE Sensor Network Protocols and Applicatons Workshop*, May 2003.

[4] K. A. Delin and S. P. Jackson. Sensor Web for In Situ Exploration of Gaseous Biosignatures. In *Proceedings of IEEE Aerospace Conference*, March, 2000.

[5] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*,2nd ed, Prentice Hall, 2003.

[6] J. Hill, R. Szewezyk, A. Woo, S. Hollar, V. Culler, K. Pister. System Architecture Directions for Network Sensors. In *Proceedings of the 0th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Cambridge, MA, Nov, 2000.

[7] C. Intanagonwiwa, 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)*, Boston, Massachusetts, August 2000.

[8] J. Liu, P. Cheung, L. Guibas and F. Zhao, *A Dual-Space Approach to Tracking and sensor Management in Wireless Sensor Networks*, Palo Alto Research Center Technical Report P2002-10077, March, 2002.

[9] S. Madden, R. Szewczyk, M. J. Franklin and D. Culler, Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. *4th IEEE Workshop on Mobile Computing Systems and Applications*, Callicoon, NY, June 2002

[10] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *the 2002 ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, Atlanta GA, September 28, 2002.

[11] R. Nowak and U. Mitra. Boundary Estimation in Sensor Networks: Theory and Methods. In *Second International Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[12] G. S. Sukhatme, D. Estrin, D. Caron, M. Mataric and A. Requicha. Proposed Approach for Combining Distributed Sensing, Robotic Sampling, and Offline Analysis for in situ Marine Monitoring. In *Proc. Advanced Environmental and Chemical Sensing Technology - SPIE 2000, Vol*, 4205 Boston, November 6-8 2000.

[13] W. Ye, J. Heidemann and D. Estrin. Sensor-MAC (S-MAC): Medium Access Control for Wireless Sensor Networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June, 2002.