Efficient Data Collection and Event Boundary Detection in Wireless Sensor Networks Using Tiny Models[1]

Kraig King[1,2] , Silvia Nittel[1,2]

[1] Geosensor Networks Laboratory
[2] Department of Spatial Information Science and Engineering
University of Maine
Orono Maine, United States 04469-5711
{kking, nittel}@spatial.maine.edu

Abstract. Using wireless geosensor networks (WGSN), sensor nodes often monitor a phenomenon that is both continuous in time and space. However, sensor nodes take discrete samples, and an analytical framework inside or outside the WSN is used to analyze the phenomenon.. In both cases, expensive communication is used to stream a large number of data samples to other nodes and to the base station. In this work, we explore a novel alternative that utilizes predictive process knowledge of the observed phenomena to minimize upstream communication. Often, observed phenomena adhere to a process with predictable behavior over time.We present a strategy for developing and running so-called 'tiny models' on individual sensor nodes that capture the predictable behavior of the phenomenon; nodes now only communicate when unexpected events are observed. Using multiple simulations, we demonstrate that a significant percentage of messages can be reduced during data collection.

Keywords: Sensors, wireless sensor network, model, continuous phenomenon, tiny models, process modeling, prediction, autonomous

1 Introduction

As the field of geosensor network research matures, the number of sensor networks deployed to collect data for geospatial phenomena is increasing. This trend is spurred by significant advances in wireless communication, the miniaturization of computing and storage hardware, as well as advances in sensor materials and technology [1]. Independent networks of sensors nodes are frequently deployed to observe and monitor the characteristics of an event. These characteristics are often comprised of dissimilar measurands of the target phenomena, which can span both 3D space and time. Consider for example, monitoring the intensity of light over a finite region, not necessarily a geographic region but for example an indoor space, which is illuminated by a controlled light source. The phenomenon of the light distribution follows an expected physical process, which can be captured in a formal model. This work

[1] Published in GIScience 2010, Zurich, September 2010, Springer Publisher.

investigates strategies to minimize data collection in the sensor network, as sensor nodes do not need to exchange information with other nodes if the predicted process proceeds as expected. Thus, instead of communicating the 'obvious', the sensor network only initiates wider-spread activity in situations where there are measured deviations from the known model (for example, if an additional light source is added).

A key challenge in developing a model-based decentralized monitoring strategy, is satisfying the storage and computing requirements that most large models will necessitate. These models should not exceed the 416Mhz and 32MB of storage provided by some of the most advanced sensor hardware offered today [2]. The goal of this research is to map a predictive, often large and complex model to a set of "tiny models", which can be run on Micaz sensor nodes with limited memory and processing resources [3]. These tiny models will provide each node with sufficient knowledge to evaluate the expected process based on time and their spatial location with regard to the phenomena being observed.

Our objective is to minimize overall data communication and reduce it to handling unexpected values. In such an event, sensor nodes will reason about the cause of the deviation; possible causes may include noisy sensor readings, an observed event, or incomplete (perhaps even inaccurate) model information..

Streams of sensor readings represent 'snapshots' of objects and processes that continually change over time. Models are generated from previous knowledge of how sensed objects and/or processes evolve over some temporal period, and are used to assist nodes in understanding and monitoring evolutionary changes in the network. An event is considered a record of a process change of interest, that is, a measured deviation from the known process model at some fixed time [4]. For example, a model may predict that light intensity at a measured location should be similar (e.g. within +/- 0.1 W/m^2) to spatially and temporally adjacent sensor readings. In this scenario, the deviation is the reported event, the location the object of interest, and the sudden change in illumination an abnormal process state. Relative to the known process model, the sensed event, object of interest, and abnormal process state become the phenomena of interest, which initiates further node communication to reason about the abnormal sensor data.

The next section introduces the research problem.. The remainder of this paper is structured as follows: a brief motivational example of how a prototypical model can assist in minimizing radio communication in a sensor network which is followed. by a discussion of the hardware constraints and operational requirements.. This will become a primer to a detailed study of methods for developing smaller models from detailed parent models that are complete and thus much larger. A formal framework and rationale is then introduced, that utilizes these 'tiny' models.We show that our strategy for decentralized areal estimation using tiny models will yield efficient, semi-autonomous sensor networks by leveraging and evolving models of an understood process on resource constrained hardware. Finally, conclusions and plans for future work are discussed in the final section of this paper.

2 Predictive Model-based Data Collection in Sensor Networks

Scientists and engineers are frequently interested in monitoring an understood phenomenon in order to verify process stability, to identify the occurrence of abnormal events, and foremost be alerted to them. For example, engineers may want to identify abnormal machine vibrations to assist in predicting mechanical failure. Today, we can use wireless sensor nodes to autonomously monitor these phenomena at novel spatial and temporal scales. However, even the most state-of-the-art hardware still has limitations such as power consumption, storage capacity, and processing capability. In particular, this applies to the application of sensor networks for the observation of well-known phenomena whose process can be captured in predictive models. Traditionally, sensor nodes are used for the raw collection of data, which is then transmitted to a central base station for verification and comparison to a forecast phenomenon. Alternatively, communication costs can be significantly reduced if the sensor nodes could autonomously and intelligently make predictions, detect abnormal values locally, and only communicate alerts in exceptional cases.

To accomplish this, we propose breaking up traditional large, complex predictive models into 'tiny models' and loading these compact models of the target phenomena onto individual sensor nodes. By executing tiny models locally on sensor nodes, it is likely that a significant decrease in communication cost can be achieved due to a reduced need for transmitting raw sensor data readings to other nodes and throughout the network. Communication activity is limited to 'unusual' events by avoiding 'obvious' data collection and instead only initiating wider-spread sensor activity where measured deviations from the known model exist. Sensor readings are autonomously and locally compared to derivations from a tiny model to isolate unpredicted observations, which are further discriminated as sensor noise, environmental noise, or an actual event of interest. For example, a sensor could measure a noisy value or indeed an (unexpected) event. Local collaboration is reduced to nodes only interacting with neighboring nodes to analyze the cause for the deviation. In the case that an event happens, likely the neighboring nodes will sense similar values, and the boundary of the 'event' can be computed.

The remainder of this paper explores various techniques for the creation and dissemination of a 'tiny model', methods for efficient boundary detection using this model, and techniques for in-network data suppression.

2.1 Example

Let's assume that the process being observed, for example the spatial distribution of light in an observed region over time, can be represented by a mathematical model of the variance in illumination intensity at any temporal snapshot T. All nodes are programmed with a minimalistic variation of this model, and once deployed, nodes persistently sample the spatial region of interest at a discrete interval that is appropriate to accurately observe estimated changes in light intensity (for example, at a frequency that is 1/100th the expected rate of change.) This observational process can be decomposed into three transition states, each of which progressively demands more node-node communication. The first state, field monitoring, is characterized by the node performing predictive sampling/validation of the environment relative to the model, requiring the least communication with neighbors. The second state, event

detection, draws upon knowledge from neighboring nodes to identify events of interest, employing a balance of internal model validation and cross-comparison with neighbors. The third state, event contour processing, requires the highest level of node collaboration to accurately monitor the event boundary.

Our objective for using tiny models is that they will permit nodes to collect data samples locally and then cross-validate this data with the model prediction. Ideally, the model will facilitate independent operation of the nodes and no radio transmission within the network would be required. However, due to spatial knowledge requirements nodes may still need to periodically communicate with each other in order to synchronize the model with reality, validate on-board sensor readings, and be responsive to neighboring requests for validation. That is, intermittent readings from neighboring nodes are necessary to refresh model predictions and also to determine if a deviation from these calculations is due to noise or some unpredicted event. In the latter case, a sensed deviation from ambient light that exceeds a predefined limit for some temporal period will trigger further processing within the network, commanding all neighboring nodes to cross-compare their sensed values of the field and collaborate to determine potential noise in the measurement or the extent of aberrations from the prediction.

To set the groundwork for a prototypical example, let's assume that twelve nodes are arranged in a fixed 3 x 4 spatial grid to monitor light intensity along the surface of a room. To simplify the illumination function, it is assumed that there is a stationary point source of light, centered in an empty room, at a fixed distance above the floor. These design constraints permit the use of a simple illumination function, which takes as input the power and location of the light source, its distance above the floor, and the measurement location. For a single light source this function is formalized with the following equation:

$$I = W / (4*pi * ((M_x - S_x)^2 + (M_y - S_y)^2 + D^2)) . \quad \textbf{(1)}$$

Where: I = illumination, Watts/m^2
W = bulb power, Watts
$M_{(x,y)}$ = Cartesian coordinates of measurement location, m
$S_{(x,y)}$ = Cartesian coordinates of bulb location, m
D = distance of the bulb above the floor, m

The formula above models illumination along the base of a room from a single source of light. For example, we consider a 60 watt bulb 3 meters above the floor and positioned at room center (i.e. $S_x = 0$, $S_y = 0$). The sensor nodes are equipped with a localized program that predicts the illumination based on the node's spatial location (i.e. $M_x = 0$, $M_y = 0$). We call this a 'tiny model' since it can be stored and computed on a resource constrained sensor node. Communication activity is minimized by locally sampling the light intensity at a discrete point and then comparing it with the internally executed model. This sampling and comparison occurs at an interval sufficient enough to accurately capture an event, for example, a sudden drop in illumination within the monitored environment. If this is the case, it initiates an in-network decentralized algorithm, which draws upon knowledge from neighboring nodes to determine the cause of the deviation, potentially isolating an event of

interest. For example, if neighboring nodes do not sense a similar deviation, it is likely that the deviant reading is attributable to sensor or environmental noise. However, if a number of adjacent nodes detect similar departures from their own tiny models, more frequent and targeted sensing within this region of interest begins and data is communicated upstream to a central node for further analysis and estimation of the event's areal extent.

2.2 Requirements

The objective of this research is to minimize communication within the sensor network by verifying process observation locally, and by leveraging knowledge that is traditionally captured in large complex models of an understood process within more compact 'tiny models'. These models need to be sufficiently constrained such that they can be executed on individual nodes within a sensor network. Characteristics that are considered include the hardware constraints of RAM, Flash, CPU, and power, as well as how system accuracy and precision are affected by design decisions such as the spatio-temporal sampling frequency of sensor data. This novel approach, known as 'TinyModeling', is expected to provide an energy reduction approaching 85% over raw data collection. This energy reduction is derived from a measured reduction in the number of messages transmitted during process observation and event detection. Message transmission is the primary consumer of a node's hardware resources, thus, a reduction in sensor network communication offers significant opportunities for resource conservation.

MEMORY/CPU FOOTPRINT: A major challenge in designing tiny models is how to capture knowledge with models that are based on point samples (or close neighborhoods) and that must run in a severely hardware constrained environment. Alternatively, traditional 'large' models run on powerful CPUs with large reference data sets, comparatively unlimited persistent storage, and substantial RAM. To better understand the operational boundaries for tiny models, one must consider the overhead required by a node operating system such as TinyOS. For example, the TinyOS kernel only occupies approximately 400bytes of storage while the required nesC runtime primitives and radio interface use another 3.1Kb [5],[6]. This lightweight operating system retains a significant portion of the sparse resources available for storage of the tiny model and associated algorithms. For example, a MicaZ mote running TinyOS provides approximately 3.7Kb of RAM, 6.1Mhz of available CPU duty, and 124.6Kb flash memory for application programming. As can be seen from Figure 2.2, on most current node platforms at least 80% of all available resources can be used for tiny models. One can also expect a significant conservation of power by implementing tiny models rather than raw data collection and communication schemes. For instance, at peak load a MicaZ node consumes approximately 19.7mA of current while transmitting and receiving messages. However, without external communication the node only utilizes 8mA of current, conserving up to 60% of the available power during deployment.

DCP RESOURCE AVAILABILITY

DCP	RAM		CPU		PROGRAM FLASH	
	AVAILABLE	REMAINING	AVAILABLE	REMAINING	AVAILABLE	REMAINING
MICAz	4Kb	3.7Kb	8Mhz	6.1Mhz	128Kb	124.6Kb
IRIS	8Kb	7.7Kb	8Mhz	6.1Mhz	128Kb	124.6Kb
Imote2	256Kb	255.7Kb	416Mhz	414.1Mhz	32Mb	31.9Mb
TelosB	10Kb	9.7Kb	8Mhz	6.1Mhz	48Kb	44.6Kb
TmoteSky	10Kb	9.7Kb	8Mhz	6.1Mhz	48Kb	44.6Kb

Fig. 2.2. Comparison of data collection platform (DCP) resource availability when running TinyOS

ROBUSTNESS/ACCURACY: The system of nodes and 'tiny models' run within a specific context: we expect to observe a likely and anticipated process most of the time, and only need to occasionally measure this process to verify that the phenomenon is proceeding according to the model. However, the nodes must accurately and quickly, detect and analyze the unexpected, classifying it as either noise or an event, and only performing sporadic sampling and verification (to avoid false positives).

The objective of developing tiny models is not to capture perfect knowledge of the observed process over some temporal period. We pose the accuracy of traditional complex models at the targeted precision, however, we assume that the compact models will be less precise. Tiny model implementation in its most compact form, must take an aggressive stance on balancing internal data collection and storage with external communication among neighboring nodes. A sufficiently significant data set, one that provides adequate temporal as well as spatial density, is required to meet the desired accuracy and precision requirements for trending and analysis. Model-based areal event detection targets the 'exceptional' case and not the 'norm', therefore, communication with neighboring nodes should only be instantiated for further clarification and examination of unexpected data. Depending on the architecture of the tiny model, periodic local collaboration between networked nodes must occur to verify sensor readings and perform model validation; however, this does not provide optimum performance. Strategies such as alternating wake and sleep cycles among nodes, as well as leveraging staggered local clocks instead of global synchronization can assist in minimizing power depletion due to communication.

3 Designing Tiny Models

The main goal in designing a tiny model is to sustain as much intra-node data processing as possible, in order to minimize network collaboration (and expensive communication) about obvious events. However, periodic node-node communication is required for model dissemination and is critical for process observation once an event has been detected. To facilitate node-node communication, an adequate network topology must exist. Although mobile WSNs offer a number of interesting challenges,

currently we focus on a static network to develop and test the TinyModeling approach. We assume a sensor network is arranged as a collection of N_i sensor nodes, located in a uniform grid pattern defined by C_x columns and R_y rows (Figure 3.0.1). In order to avoid orphaned nodes, it is assumed that each node is positioned so that it can collaborate with at least one neighbor. That is, the distance between spatially adjacent nodes must not exceed the reliable transmission range of the hardware being used. Such a topology simplifies the algorithms used for location aware models by minimizing distance variation in the spatial distribution of sensors, allowing the communication paths between nodes to be predictable.

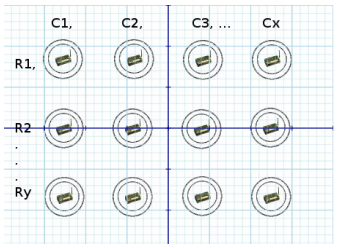

Fig. 3.0.1. A prototypical sensor network composed of C_x columns and R_y rows.

Using this topological structure as our test bed for deployment of up to 289 sensor nodes, we introduce a first mechanism for disseminating prior knowledge of a phenomenon (e.g. the illumination model discussed earlier) to individual sensor nodes within the network. Overall, our interest is in 'mapping' large complex models into tiny models that can be run on individual sensor nodes. In the following research, we test our approach hypothesis using a well-understood physical process (light distribution from one or more light sources) that we can capture in an equation and simulate as a process. The objective is to compute the overall model, initialize the sensor network with these tiny models, and then test the model-based process (observation). Overall, the experiment consists of several steps: (1) setting up a communication topology to initialize the nodes with the tiny models, (2) initializing individual nodes with their tailored tiny models, (3) process observation and (4) in-network localized event handling.

Starting with step 1, a single node is designated as the base station (i.e. root) and it is preloaded with the large complex model of the phenomenon being observed. It is assumed that the base station node is a line-powered device, connected directly to a PC via a physical connection such as serial or USB. At initialization of the sensor network the root coordinates communication between all nodes, helping to assert the routing protocol that will be used for future upstream and downstream communication among nodes. A discussion of these communication strategies is outside the scope of this paper, however, we encourage readers to review established techniques such as tree, star, and clustered topologies. For this research, the multi-hop tree collection schema provided by TinyOS has been utilized to flow messages to and from the root.

Upon successful self-organization of the sensor network, with regard to the initialization routing tree, the root node queries the spatial location of all other nodes. These coordinates are fed into the larger, complete model, with the resulting output

being the coefficient(s) describing the phenomenon at the prescribed node location. This information becomes the basis for a 'tiny model', which will be transmitted back to the node and stored for future in-network sensor validation. For example, let's assume a node at location (2, 4) transmits its position to the root of the network tree. The root then calculates the intensity coefficient of the light at the node's reported location. Assuming a ninety watt bulb located at the center of the room (0,0) and positioned three feet above the floor, the calculation per equation **(1)** is:

$$I = 90 / (4*pi *((2 - 0)^2 + (4 - 0)^2 + 3^2)) .$$
$$I = 0.247 .$$

The resulting coefficient is transmitted back to the node and becomes the basis for the most simplistic type of tiny model, a single coefficient, which describes the anticipated sensor reading at the current measurement location. This process repeats iteratively until each sensor node has received its own miniaturized version of the larger model from the root node (albeit in this case only a coefficient). Thus, this framework permits model prediction at all node locations.

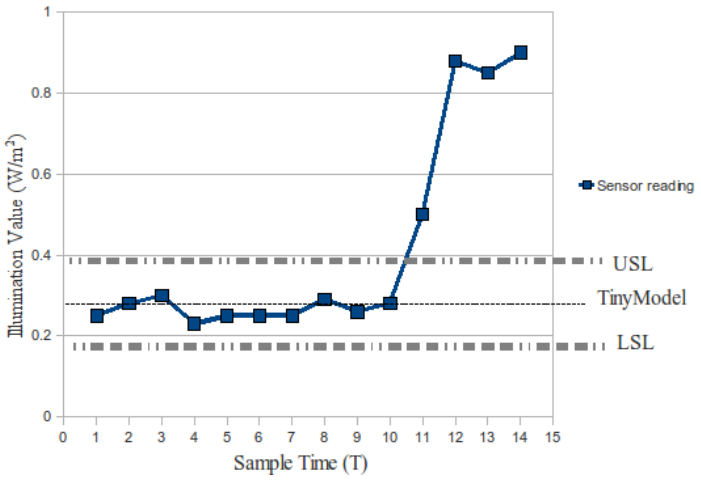

Fig. 3.0.2. The graph above demonstrates sensor values and their various states of adherence to predictions made by a tiny model of the observed phenomenon, represented here, by the dotted line at the ordinate 0.3. Thus the process can be described as (a) compliant, readings 1-10 and (b) uncertain, readings 11-14

Upon receipt of the tiny model definition, phase 3 starts, and each sensor node is released to begin internal data acquisition and autonomous comparison to the predicted nominal value. Should the sensor value remain within a predetermined upper and lower tolerance band (i.e. upper spec limit, USL; lower spec limit, LSL) about this nominal, the node's radio remains off and it is assumed by the root that the process (e.g. illumination) remains compliant (Figure 3.0.2a). Alternatively, the sensed value may drift outside of the permissible tolerance thresholds, indicating the

presence of data uncertainty or the occurrence of some event (Figure 3.0.2b). Before either explanation of variation is considered plausible, additional analysis must first occur to deduce the cause.

4 Event Detection Based on Tiny Models

During the comparison of sensed data to the tiny models' prediction, nodes must be empowered to classify the detected result as either: (1) a non-event, i.e. the process behaves as expected, (2) uncertainty due to environmental noise, (3) faulty readings due to sensor failure, or (4) an event that requires further action to isolate its boundary. We define an event explicitly to be drift from the predictive model, which is characterized by a discrete spatiotemporal dimension. This value is defined by the system user, and quantifies the threshold between noise and event detection. For example, if the sensed illumination suddenly exceeds a permissible threshold (e.g. the upper specification limit -USL) beyond the model prediction, the node must have the necessary logic and resources to reason about the cause of the drift. Possible events may be an object passing through the network, which induces a shadow, or the introduction of a second light source, which increases the light intensity at specific node locations.

Once a significant departure from the tiny model has been identified, the node attempts to mitigate the cause autonomously. This is achieved by comparing the noncompliant sensor reading to an internal cache of historical sensor readings: $\{R^{-1}, R^{-2}, R^{-3}, \ldots R^{-n}\}$. The depth of the cache, **n**, must be sufficient enough interpolate trends which may be indicative of an event, but not so large that it consumes excessive hardware resources. This system parameter is highly dependent upon the temporal sampling schema, as well as the expected rate of change of the phenomena being measured. For instance, the illumination example may be understood to react as an approximate binary process, with a rapid and near constant drift from the expected value (e.g. a new light source is suddenly added or the original one goes dark). In such a scenario, a cache size of ten historical values may provide sufficient resolution to flag a recurring departure from nominal that requires additional investigation.

For example, if the current sensor reading $\{R^{0}\}$is non-conforming but the previous ten readings $\{R^{-1}\ldots R^{-10}\}$ match the prediction, the node assumes the uncertainty is attributable to environmental noise and no further action is taken. However, should a statistically significant number of these historical readings also exhibit a similar departure from nominal, further processing will be initiated to identify if the cause is a faulty sensor or the existence of an event. Because the node has no additional internal knowledge available for reasoning, it must initiate radio communication within the sensor network to query if neighboring nodes have experienced similar departures from their own model predictions. If no neighboring nodes detect such departures from the model, the node assumes that the abnormal reading is due to a faulty sensor. It continues to analyze future data acquisitions, but waits a defined number of measurement cycles before it again queries neighboring nodes for the existence of confirmed model departures.

Alternatively, neighboring nodes may return information that they *have experienced* a similar model departure. When such a confirmation occurs, nodes self-organize to detect the extent of the occurring event. Based on previous work [7],[8] an energy efficient algorithm is used to identify only the boundary of the areal event and track its changes over time. Each node communicates with its direct neighbors to identify if it is located on the boundary of the event or "inside" of the event. In the case of being a node located 'inside' of an event's region, all neighboring nodes show similar derivations from their models. In this case, the node stops communication again, and resumes regular local sampling. If a node, however, identifies that several of its neighbors experience no model derivation, but others do, it can identify itself as a boundary node and identify potential other neighboring boundary nodes.

Once all sensor nodes have identified themselves as either boundary nodes or 'inside' event nodes, the boundary nodes continue communication in regular intervals to observe the event boundary and its changes. Successive monitoring of the event boundary is only performed by these boundary nodes, and the one closest to the root is elected to transmit an aggregated list of the nodes which compose the event boundary. This decentralized algorithm permits efficient boundary estimation, minimizing the number of messages required to monitor the event. To improve the resolution of the sensor network, the boundary nodes may also increase their sensor sampling frequency in order to monitor the event with a higher precision. Should any node begin acquiring a consecutive number of sensor readings that are within normal operating parameters, it will stand down, and cease to transmit data until a future event is sensed (using the procedure prescribed above). This protocol permits nodes to minimize unnecessary radio communication, by only transmitting data when a confirmed event has been detected.

5 Performance Evaluation

The tiny model framework consists of several algorithmic parts: (1) initializing the nodes with their models, (2) continuous observation, (3) noise, event detection and identification. Since the initialization is run only once, or rarely (e.g. recalibrate), we assess the communication cost for this part separately. The major part of the performance testing is done with regard to steps two and three. We expect that phenomenon observation without event detection is the most interesting part of the performance analysis since we foresee the highest energy savings here. Event detection and handling is similar to other approaches in this research area (e.g. boundary detection algorithms); however, it is performed based on tiny model information. In our experimental set-up we test the communication cost for all three parts, and compare the observation with raw sensor data collection and tree based routing as a baseline.

For simulation and testing purposes, three 4m by 4m grid topologies of different sensor node 'resolution' were constructed (see Figure 5.0.1). The first is composed of 25 nodes with a 2m spacing. The second contains 81 nodes with a 1m spacing and the most dense configuration contains 289 nodes with a 0.5m spacing. Each of these topologies has a relaxed signal to noise schema to facilitate predictable

communication (i.e no delayed or lost messages) among all nodes in the network. This eases retransmission of messages due to signal variance and path loss, permitting a more accurate performance evaluation of event detection utilizing the tiny model framework. The varying choices of sensor network density are necessary to assess the detection accuracy.

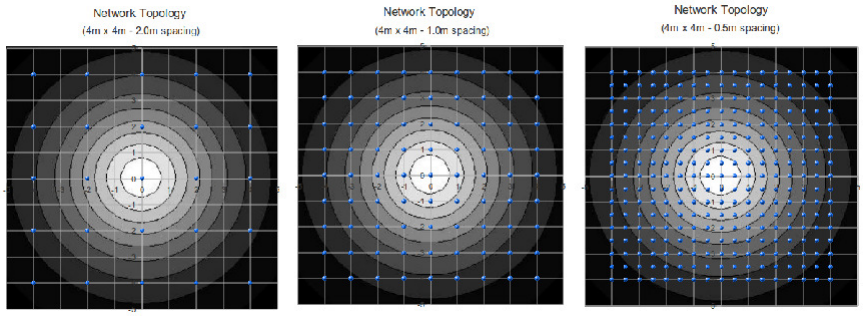

Fig.5.0.1. Prototypical sensor topologies used to measure the illumination gradient for a light source located at (0,0). (a) 2m spacing, (b) 1m spacing, (c) 0.5m spacing.

To organize the nodes, a multi-hop, tree topology-based network configuration is employed. The simulation is executed for twenty-four hours with sensor data sampling occurring at five minute intervals, testing the three different network densities. In the case, of raw sensor data collection, the sensor data is sampled and sent upstream to the base station without further aggregation. Analyzing tiny models, we test several sensing strategies to quantify the number of messages TinyModels require, and compare this with the number of messages in the raw sensor data collection case.

In the *first test*, we determine the simulation baseline for the TinyModels approach; here, we assume the tiny models are disseminated to the nodes once, and no further events, noise or even synchronization takes place. In the *second test*, we add a synchronization beacon to the TinyModel protocol, requiring all nodes to transmit a confirmation of functionality to the root every hour or two hours. In the *third test*, we additionally factor in that a percentage of the nodes experience sensor noise locally, which is characterized by internal sensor data exceeding model predictions but having no correlation with readings from neighboring nodes. This noise occurs for a finite period of time, at a frequency of once per hour. More specifically, scenario 3a tests 20% of the nodes experiencing noise while test 3b five assumes 40% of the nodes experience noise. In the *fourth test*, we select a set-up that uses hourly synchronizations of nodes with regard to the tiny models and a 20% noise ratio. It should be noted that unclassified noise, which has both a spatial and temporal extent may be misconstrued as an event if clusters of nodes experience similar noisy data. Such false positives for event detection require the system user to later classify these occurrences as either events or noise. Additionally, we assume event detection over a discrete spatial area within the sensor network covering about a sixteenth of the overall observation area. We assume that three events are detected (at hours 6, 12 and 18). It takes one cycle to detect an event, and can take several cycles to observe the

event depending on its duration. In the simulation, the 6th hour event is detected during a single five minute collection cycle; event detection spans four collection cycles during the twelfth hour, and two collection cycles during the eighteenth hour. The results of each simulation are shown in Figures 5.0.2 and 5.0.3.

These test scenarios demonstrate that the TinyModel approach significantly reduces sensor network energy consumption during routine monitoring of a well understood phenomenon. Most notably, the number of messages for the baseline scenario of raw data collection in a 289 node topology was reduced from 521,280 to 46,583 messages using TinyModel-based event detection. This is due to the dissemination of intrinsic process knowledge that empowers network nodes to autonomously reason about the sensor data they acquire. The creation of tiny models, coupled with targeted distribution of this knowledge, enable in-network data evaluation that minimizes radio communication with the root node when the process is operating as expected. The results for each simulation and the associated reduction in messages are shown in Figure 5.0.4.

Based upon these results, TinyModel-based event detection benefits mostly those processes which are assumed to be relatively stable and that frequently operate within the prescribed specification limits of the model. Given such a process, tiny models significantly reduce overall network communication, permitting node radios to remain off, conserving both power and hardware resources.

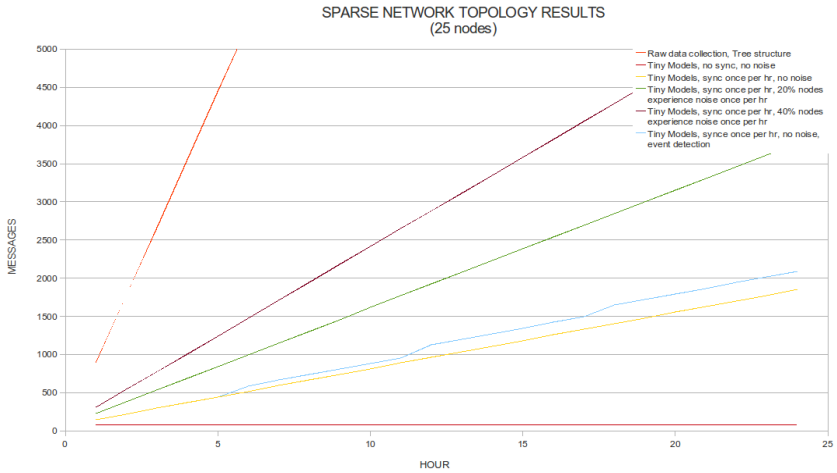

Fig. 5.0.2. Raw data collection vs. TinyModeling test results for a sparse topology (e.g. 25 nodes).

To enable more extensive testing of the hypothesis that TinyModels for a known process significantly decreases communication costs during event detection, we have developed a prototypical sensor network using TinyOS and the TOSSIM simulator. This system is sufficiently modular that additional knowledge models can be inserted into the simulation, as well as various communication protocols, and sources of noise. Additionally, the temporal frequency and spatial extent of events can be altered to test the robustness of the TinyModel framework. It is anticipated that continued testing

and development will facilitate additional enhancements, which will further improve the performance of TinyModel event detection.

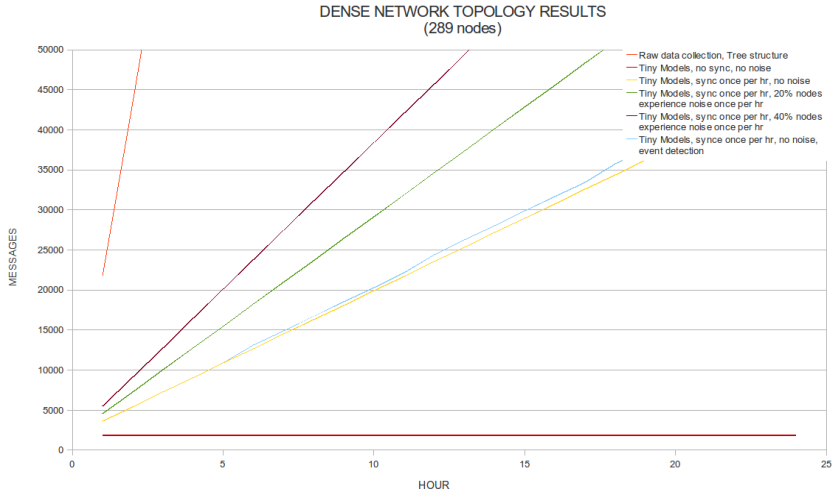

Fig. 5.0.3. Raw data collection vs. TinyModeling test results for a dense topology (e.g. 289 nodes)

NUMBER OF MESSAGES TRANSMITTED (24hr simulation)						
	25 NODES		81 NODES		289 NODES	
	messages	reduction	messages	reduction	messages	reduction
BASELINE, raw data collection	*21312*	-	*105408*	-	*521280*	-
TinyModel, no sync	74	-99.65%	366	-99.65%	1810	-99.65%
TinyModel, with sync	1850	-91.32%	9150	-91.32%	45250	-91.32%
TinyModel, with sync, 20% noise	3770	-82.31%	15371	-85.42%	67445	-87.06%
TinyModel, with sync, 40% noise	5690	-73.30%	21592	-79.52%	89640	-82.80%
TinyModel, with sync, events	2091	-90.19%	9659	-90.84%	46583	-91.06%
	AVG:	-87.35%	AVG:	-89.35%	AVG:	-90.38%

Fig.5.0.4. Simulation Results: an analysis of messages transmitted per 24 hour simulation for each of six testing scenarios and three network topologies.

6 Related Work

Due to a node's limited power capacity and the high cost associated with wireless data transmission, improving communication efficiency between networked sensor nodes has been an active area in geosensor research [9]. Many strategies have been introduced to make geosensor networks more efficient by decreasing node-to-node communication and the associated transmission costs. For example, data-centric routing, draws upon an in-network analysis of individual sensor readings to permit nodes to evaluate whether or not sensed data should be sent or received [10]. For

instance, nodes may choose to only power on their radios if a message of interest (e.g. that of a detected event) should be transmitted to neighboring nodes [11].

Models of an understood process are also an integral part of ongoing research that aims to advance structural health modeling using sensor networks [12]. In this application, groups of sensors are distributed throughout an engineered structure (e.g. building or bridge) to monitor vibrations that may compromise the structure's safety or useful life. Groups of sensor nodes are strategically paired to process a single structural analysis algorithm in a coordinated manner. Nodes locally process sensor readings and the collective leader transmits an aggregated set of data, or recommended model adjustment parameters, back to a reference node. Our Tiny Model strategy further decreases the volume of messages exchanged between nodes by empowering each node to autonomously analyze sensor data, and only initiate radio communication if values deviate from known model predictions.

Another field of geosensor research, tracking the patterns of moving objects, utilizes models to efficiently track the current motion of an object as well as predict future movement of the objects. One such example leverages materialized and non-materialized trajectories to improve sensing efficiency and overcome location imprecision due to uncertain data.. In addition to using the road network as a source of knowledge about the phenomena being measured, more robust moving object modeling techniques also consider velocity changes of the moving objects being sensed [12].

Within the sensor network and database communities, models have been proposed to assist with user-based queries for data acquisition in sensor networks [13]. Sensor readings are supplemented with knowledge from predictive approximation models to augment the need to collect data from all sensors within the network. These strategies typically use statistical modeling techniques to account for issues in spatial sampling by extrapolating missing or faulty sensor data. In this paper, instead of model predictions being generated by a central coordinator and issued to downstream child nodes, we propose empowering all nodes with a miniaturized version of the process model..

7 Conclusions and Future Research

This research is different from previous work in the area of efficient sensor networking in that it utilizes localized tiny models to perform energy efficient data collection and boundary analysis of events, by leveraging characteristics of an understood phenomenon to achieve process monitoring and unexpected event detection similar to that realized with a fully detailed model. Comparing locally sensed data to the tiny models' prediction, nodes must be empowered to classify the detected result as either: (1) a non-event, i.e. the process behaves as expected, (2) uncertainty due to environmental noise, (3) faulty readings due to sensor failure, or (4) an event that requires further action to isolate its boundary.

When a process behaves as expected, nodes only perform predictive sampling/validation of the environment relative to the model, requiring the least communication with neighbors. If deviation from the model occurs, nodes employ a

balance of internal validation and cross-comparison with neighbors to isolate sensor noise and detect event boundaries. Using a simulation approach, this work has demonstrated that 'TinyModeling' is able to provide an energy reduction exceeding 85% over raw data collection (measured in transmitted messages). Additional testing will quantify the accuracy and precision of boundary detection for events, by varying the spatial density of the sensor topology and the temporal sampling schema. Future research will consider methods for achieving immunity to sensor failure, reasoning about abnormal event detection, model-evolution using back propagation, and the implications of mobile nodes.

References

1. Nittel S.: A survey of geosensor networks: advances in dynamic environmental monitoring. Accepted for publication: Sensors Journal. (2009)
2. CrossbowTech: Imote2. http://www.xbow.com/Products/productdetails.aspx?sid=253. Visited 10.04.2009.
3. CrossbowTech: Micaz. http://www.xbow.com/Products/productdetails.aspx?sid=164. Visited 10.04.2009.
4. Galton A and Worboys M.: Processes and events in dynamic geo-networks: In:Rodriguez, M., Cruz, I., Levashkin, S. & Egenhofer, M. J. (eds.): Proceedings of the First International Conference on Geospatial Semantics, GeoS 2005. Lecture Notes in Computer Science (LNCS) Springer Verlag, Berlin. Volume 3799. Pp 45–59. (2005)
5. Hill J, Szewczyk R, Woo A, Hollar S,Culler D, Pister K.: System architecture directions for networked sensors: ACM SIGPLAN Notices. Volume 35, Issue 11. Pp: 93-104. (2005)
6. Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A, Gay D, Hill J, Welsh, M, Brewer E, Culler D.: TinyOS An operating system for sensor networks: Ambient Intelligence, Springer Berlin Heidelberg. Volume 2. Pp: 115-148. (2005)
7. Duckham M, Nittel S, and Worboys M.: Monitoring dynamic spatial fields using responsive geosensor networks: ACM-GIS 2005, Bremen, Germany. (2005)
8. Jin G and Nittel S.: Supporting spatio-temporal queries in wireless sensor networks by tracking deformable 2D objects: ACM-GIS 2008, Los Angles, CA. (2008)
9. Stefanidis A., Nittel S: GeoSensor Networks: CRC Press, Boca Raton. Pp: 296. (2005)
10. Huang H, Hartman J, and Hurst T.: Data-centric routing in sensor networks using biased walk: 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks(SECON '06). Volume 1. Pp: 1-9. (2006)
11. Ditzel M and Langendoen M.: D3 Data-centric data dissemination in wireless sensor networks European Conference on Wireless Technology, Paris, France, October 2005. (2005)
12. Nagayama T, Spencer B, Agha G, Mechitov K.: Model-based data aggregation for structural monitoring employing smart sensors: Proceedings of the Third International Conference on Networked Sensing Systems (INSS 2006), May 31- June 2, 2006. Pp: 203-210. (2006)
13. Deshpande A, Guestrin C, Madden S, Hellerstein J, Hong W.: Model-driven data acquisition in sensor networks: Proceedings of the 30th International Conference on Very Large Databases (VLDB). Toronto Canada, Volume 30. Pp: 588-599. (2004)