THRESHOLD-BASED ALGORITHMS FOR POWER-AWARE LOAD BALANCING IN SENSOR NETWORKS

Christopher M. Cianci, Vlad Trifa, and Alcherio Martinoli

Swarm-Intelligent Systems Research Group, École Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland. {Chris.Cianci, Vlad.Trifa, Alcherio.Martinoli}@epfl.ch

ABSTRACT

Given the rigid energetic constraints under which a sensor network must operate, efficient means of power management are vital to the success of any sensor network deployment, particularly those in rapidly changing environments. Threshold-based algorithms provide a possible in-network method for adaptive distributed control of energy consumption.

1. INTRODUCTION

In the rapidly expanding field of sensor networks, power management is of paramount importance. The highly distributed nature of relevant applications often renders wired deployments infeasible, and frequent battery replacement in a deployed network is unlikely to be practical due to the time and effort required to perform such a task. Therefore, until the power requirements of the hardware and the ability of a node in the network to harvest power from its environment converge, any battery-operated node will necessarily have a finite lifetime—which one obviously wants to be as long as possible.

More specifically, the quantity of interest is the lifetime of the network as a collective entity, and the resources available for optimizing this lifetime are the individual nodes and their respective available power reserves. While one can envision various methods for approaching this problem, the common element will usually involve taking advantage of the inherent redundancy in the distributed system and avoiding duplication of work where and whenever possible. In this context, the question then becomes one of how to assign tasks (distribute the workload) in such a way that duplicate computations are minimized while system integrity is maintained. Accomplishing this in a distributed manner, particularly in a rapidly-changing environment, can be difficult, requiring the network to detect and adapt to the current circumstances while operating.

This type of load balancing, through the mediation of redundancy, has been previously addressed in terms of routing [5], sensing [4], and other individual elements of a sensor network. Here, we explore the possibility of informing and influencing this type of decision in a distributed manner using real-time local estimates of remaining energetic resources.

1.1. Swarm Intelligence (SI)

Swarm-intelligent principles were originally inspired by observation of various natural phenomena, including the collective behavior of social insects and flocking and shoaling in vertebrates. However, due to significant hardware differences between natural and artificial systems, the swarm-intelligent metaphor is continuously being readjusted in order to fit the relevant natural and artificial technological constraints.

The application of SI to distributed, real-time, embedded systems aims at developing robust task-solving methodologies by minimizing the complexity (including the intelligence) of the individual units and emphasizing parallelism, and self-organization.

From an engineering standpoint, the principle advantages of swarm-intelligent system design are four-fold: *scalability*, from a few to thousands of units; *flexibility*, as units can be dynamically added or removed without explicit reorganization; *robustness*, not only through unit redundancy but also through an adequate balance between explorative and exploitative behavior of the system, and *simplicity* (and low-cost) at the individual level, which also increases robustness.

1.2. Threshold-Based Algorithms

Many *swarm-intelligent* algorithms have been patterned after the behavior of social insect societies. One such example is the class of "threshold-based" algorithms.

Threshold-based algorithms model group behavior based on a small number of control parameters (thresholds) that affect whether or not a particular task will be executed by a given swarm member based solely upon the intensity of the stimulus presented in comparison to the threshold currently assigned to the individual in question. In this way, decision-making is fast and reactive. The response to a stimulus can be deterministic or probabilistic depending on the amount of randomness the designer wants to introduce into the task-allocation algorithm. Heterogeneous thresholds, or thresholds which are allowed to change and become heterogeneous over time as a function of stimuli encountered and tasks performed, can lead to specialization, and division of labor [3].

1.3. Foundations and Prior Related Works

The apparent similarities between a community of social insects and a swarm of mobile robots have inspired a variety of research efforts in applying the observed behavioral model of the insects to the design and analysis of control strategies for the robotic swarm. For instance, threshold-based algorithms have been successfully applied to division of labor schemes in swarm robotics ([1], [2], [7]) yielding interesting results.

Noting the similarities between swarm robotics (or more generally, 'sensor and actuator' networks) and sensor networks, our work explores the application of swarm-intelligent algorithms to the control of sensor networks. Specifically, this paper proposes a feasibility study on the optimization of network power consumption modeled on threshold-based algorithms, which have been used quite successfully in multi-robot systems. We would like to determine whether or not such algorithms are applicable and interesting in the context of sensor networks.

2. EXPERIMENTAL APPROACH

Our focus is primarily on densely-packed event-based applications; an area which we feel has not yet been well explored. Rather than periodically sampling a continuously available phenomenon, such as temperature or humidity, our model looks toward monitoring events which may be sparse in time, and in situations where adequate prediction of occurrences may not always be possible. Therefore, under these conditions, the treatment of the network as a single fully-connected graph covering the detection area is reasonable, albeit naturally leaving open the possibility of future work extending the algorithm to larger spacial distributions (and implicitly less than complete connectivity).

2.1. Case Study

For the current work, a simple acoustic monitoring task is considered; a sensor network is asked to observe a binary source (a tone generator with only two states: on/off) in two dimensions (the sensor nodes and the source all lie in the same plane). Events travel at the speed of sound, and are attenuated in the atmosphere. For simplicity, reflections are ignored in the initial analysis, and communication is approximated as instantaneous, reliable, and radially symmetric. The behavior of each sensor node is governed by a simplistic finite state automaton. An event is considered to be detected by a node if the node is awake (listening) while it occurs; similarly, if the node is sleeping for the entire duration of an event, that event is missed (by the node in question).

Depending upon the spatial distribution and the synchronization (or lack thereof) of the nodes, different sets of nodes will be awake and listening at different times. In this paper we consider a sound source with two types of temporal patterns: periodic and aperiodic.

In order to provide a realistic basis for our calculations, we have used the logical structure and measured electrical characteristics of the MICAz sensor network platform (the newest member of the MICA [6] family). Typical measured values of current consumption by section/task are shown in Figure 1.

Sleep	I_s	528 μA
Idle	I_i	3.98 mA
Processing	I_p	8.02 mA
Radio On	I_r	20.89 mA
Transmitting	I_t	23.32 mA

Figure 1. The MICAz module with antenna, and table of average measured power consumption per activity (samples of such measurements can be seen in Figure 5). Note that powering the radio, such that messages may be received, is particularly expensive, regardless of whether or not any messages are actually received.

Simulation using these measurements estimates mean energy consumption per node that is active, with its radio on, over the course of 24 hours to be about 1.5J; at this rate the network would have a functional lifetime of around five days. However, seeing as the network is highly redundant, for all of the desirable reasons previously mentioned, one can imagine a drastic energy savings by putting various nodes to sleep at different times.

2.2. Characteristics of the Solution Space

Since a deliberative centralized scheduler would be complex, scale poorly, and represent a single point of failure in the system, we are principally interested in exploring possible distributed solutions.

As in any system, the complexity and even the fundamental nature of the solution is heavily influenced by the amount of information available a priori about the task; in the case of monitoring tasks such as this one, it will vary as a function of information available to us about the entity or entities to be monitored.

For example, if we know beforehand exactly what the actions of the event source will be (strictly periodic emissions, or otherwise according to a known schedule), we can simply instruct the network to only activate when the events are going to occur, and sleep for the intermediate times when nothing of interest is happening. Alternatively, events might be known to arrive with a frequency that is only approximately regular, which would require the network to behave in a slightly more intelligent fashion. If, however, we know little or nothing about the source (perhaps the emission times are completely random), the corresponding network controller will have to be more flexible, and it may not even be possible to engineer an efficient solution, in which case it may be necessary to invoke a learning algorithm.

Our detection task can be approached from two sides: coverage (ensuring that, with high probability, a quorum of properly functioning nodes is active at any given point in time) and adaptation (allowing a node, or even the network as a whole, to sleep during times when it feels that events are unlikely to occur). Prediction/adaptation can be very interesting when the event source being monitored is to some degree predictable. However, so as not to restrict ourselves to predictable sources, our primary focus will be on the first approach, coverage.

2.3. Engineered Controller

Having selected the coverage task, we must ensure that at least one node is active at any time during the course of the deployment. Since the objective is to minimize power usage, the best performance will result from having *only one* node active at a time. Clearly, for a task more involved than simple detection (such as acoustic localization), or in an environment where measurements are less certain, it may be desirable or even necessary to keep more than one awake at a time; this analysis is being covered in our future work.

A naïve engineered solution might be to simply specify that they take turns being on active watch, in the order specified by their id. But this poses at least two sizable problems, namely that it requires every node to (at the minimum) know the total number of nodes in the network, and that a single malfunctioning node would have the potential to result in complete network failure. In the absence of difficulties, though, some basic analysis of this type of controller can provide us with a baseline measurement of the most efficient performance we might be able to expect from our other solutions.

2.4. Threshold-Based Controller

Since the principle concern here is energy consumption, a smarter, more effective, and scalable mechanism could be to consider the energetic resources (remaining battery level) of the individual nodes during the shift assignment. As a possible solution, we propose a threshold-based algorithm, which allows each node in the network to carry out its decision-making process in a fast and effective way.

In order to accomplish this, we need to first draw a mapping between our physical setup and the terminology of the threshold model. The model depends on three crucial elements: the *stimulus*, the *response*, and the *threshold* (which controls when the response is performed relative to the presented stimulus).

Based on the success criterion of the task being considered, one might think that the "number of missed events" could constitute an effective stimulus, as when more events are being missed, we probably want more nodes to be called into action. However, there is a fundamental problem with this proposal, as neither at the individual level nor at the group level does any member or part of the network have any idea what the value of this stimulus might be (clearly, since if the events are undetected, by definition, the network doesn't know about them). And while this task might be performed by a supervisor, that would bring us back into the realm of centralized control (not to mention being impractical outside of simulation).

Therefore, the *stimulus* must be something that can be

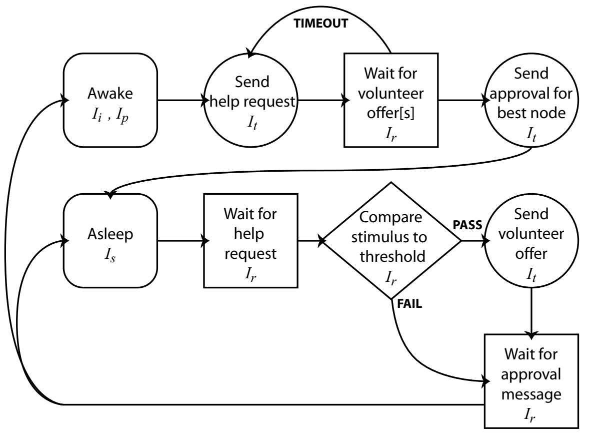
TIMEOUT
Awake I_i , I_p
Send help request I_t
Wait for volunteer offer[s] I_r
Send approval for best node I_t
Asleep I_s
Wait for help request I_r
Compare stimulus to threshold I_r
PASS
FAIL
Send volunteer offer I_t
Wait for approval message I_r

Figure 2. Finite state machine describing the behavior of a single node participating in a shift change. Current draw is given in terms of the I values from Figure 1. Messages are slotted to avoid collisions. The rounded rectangles (Awake and Asleep) last a length of time determined by the shift length. Circles are short fixed-length phenomena (like transmitting a message). Squares represent blocking wait states.

measured locally, and the *response* something that can be *done* locally (or not, depending on the threshold).

To accomplish this, we propose the following: upon initialization, one node is selected (perhaps at random) to take the first watch. At the end of its shift, the node will broadcast a message asking for a volunteer to take over the watch. Any node that hears this message will compare the stimulus to its threshold, and if the stimulus exceeds the threshold, will respond that it is available to take over the watch. The requesting node will acknowledge one offer (for instance, the first one it receives) and go to sleep, while the volunteer is now on active duty. At the end of this shift, the new active node will repeat the procedure, passing control to another volunteer. [1]

The shift length must be agreed upon within the network, as unless they all activate their radios at the same time, they will be unable to communicate with each other. While for these tests we arbitrarily set it to a constant, there may be an optimal value taking into consideration the features of the coverage task and and the sensor network system, but tests exploring this dimension of the solution space will occur in future work.

So now we have specified where the stimulus fits in, but we still have a little bit of freedom in terms of how it is actually calculated. Usually, threshold-based decisions are probabilistic; the threshold is used to position the inflection point of a sigmoid function that, for an input stimulus, outputs the probability with which this agent will be willing to perform the task. However, because of the computational constraints of the agents, it is advantageous for us to minimize the amount of calculation necessary. We can achieve nearly the same results by utilizing a deterministic response to a random stimulus generated by the node seeking volunteers; as if the probabilistic threshold response were described by the Heaviside function $H(\sigma - \theta)$, where σ, θ represent the stimulus and the response, respectively (see Figure 3). In this case, the generated stimulus is drawn from a uniform distribution on $[0, 1]$, the same range as the possible threshold values.

However, this still leaves us with the possibility that the randomly generated stimulus may be too low to provoke volunteers. If no other nodes volunteer, the stimulus will be increased (by a deterministic amount; an algorithmic parameter), and the request will be resent. Eventually, for instance, when the algorithm is extended to a larger spacial distribution of the network, we think it would be interesting to see the value of the stimulus affect the transmission power with which the request is sent, and consequently the range (local neighborhood) that the request is sent to. No response would prompt an increase in the stimulus; a louder request that can be heard by more nodes.

[1]The node that performed the previous shift is also eligible to respond to its own request, if its remaining energy places it still within the set of most desirable candidates.

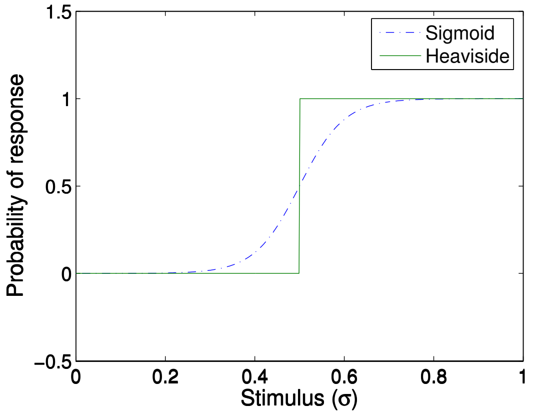

Figure 3. Probability of response versus stimulus for a node with threshold $\theta = \frac{1}{2}$, using sigmoid or Heaviside behavior.

Notice also that if the thresholds are fixed, it will nearly always be the case that the nodes with lower thresholds are used to extinction before the nodes with higher thresholds will allow themselves to be consistently conscripted. But, if each node is able to adapt its threshold as a function of its remaining energy reserves, we should be able to distribute the work load more evenly, achieving a more balanced solution.

A further solution we plan to investigate in the future is to generate the randomization of the response of a given node based on the receiver threshold randomization. In other words, while the stimulus emitted by a given leader coordinating a shift change will be completely deterministic and a function of its own battery level (with possible increments due to lack of responses from other nodes), the threshold of a given receiving node will not only be a function of its battery level but will be perturbed by a given amount of uniformly distributed noise.

2.5. Microscopic Modeling

Similar to [8] and [9], we use a time-discrete microscopic model for simulating the dynamics of the whole system. In this model, each real node is represented by its corresponding Finite State Machine (FSM, see Figure 2) where each individual state is characterized by a given power consumption.

Tests were performed to compare the different algorithms discussed in a simplified simulated environment built using C++ and Matlab. Parameter values and power consumption estimates were initially based on the MICA2 measurements reported in [10], and eventually replaced with similar measurements that we have taken here on the MICAz (see Figures 1, 4, and 5).

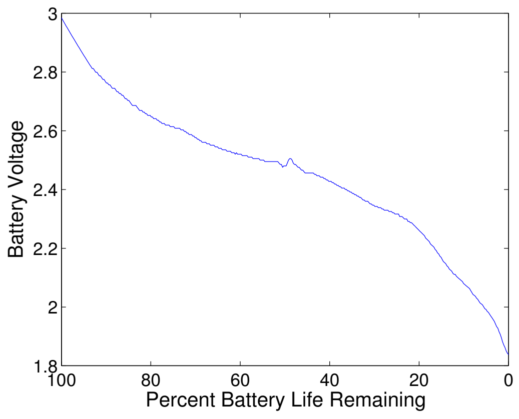

Figure 4. In-situ voltage measurements from the MICAz, over the lifetime of a pair of AA batteries.

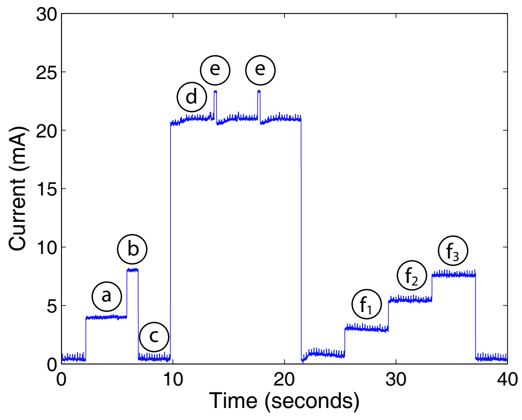

Figure 5. Measured current draw of an operating MICAz node. (a) processor idle, (b) processor active, (c) sleep, (d) radio receiver on, (e) radio transmitting, (f) LEDs [one, two, and three, respectively]. (See also the numerical values in Figure 1.)

2.6. Implementation and Testing in Hardware

A testbed based on this algorithm has also been successfully implemented on a physical network of MICAz sensor nodes.

These tests were done on a network of nine nodes. A tenth non-participating node was used as a base-station to log the negotiation messages sent within the network. For stress testing, and to shorten the experimentation time, the time between shift changes was fixed at 20 seconds, and the active node executes a mathematical transform (cpu-bound process) on detected events, requiring approximately two seconds. These additional loading factors will allow us to show that the workload is indeed shared relatively fairly among the nodes in the network, but as the time spent in the negotiation phase is non-negligible with respect to the shift length, projections about actual network longevity based on these tests are not expected to show particularly long lifetimes.

3. TESTS AND RESULTS

Here we provide and evaluate various simulated and physical implementations of the proposed algorithm, as described above.

3.1. Performance Metric

Before conducting tests, it is necessary to establish a means for interpreting their results. For purposes of evaluation, we use a network performance function of the following form to compare different control strategies:

$$M = \alpha_1 \left(1 - \frac{\frac{1}{N}\sum_{i=1}^{N} \hat{E}_i}{\hat{E}_{max}}\right) + \alpha_2 \left(1 - \frac{\max(\hat{E}_i) - \min(\hat{E}_i)}{\max(\hat{E}_i)}\right) \quad (1)$$

where N is the number of nodes in the network, the α_j are weighting coefficients ($\sum_j \alpha_j = 1$, see Figure 6 for examples), and

$$\hat{E}_i = \sum_{k=0}^{K} E_i(kT). \quad (2)$$

$T = 1$ms is the quantum of time-discretization, and $K = (1000 \cdot 3600 \cdot 24) = 8.64\text{e}7$ is the total number of iterations. To properly normalize the first term, we need to consider the maximum energy a node could possibly expend (see section 3.2.1 for the calculation of a value for $\hat{E}_{max}$).

This gives us a quantifiable measure by which to evaluate the performance of a control strategy by explicitly rewarding characteristics we consider desirable, and punishing those which are not. In this case, our goal is to minimize the amount of energy consumed (first term) while simultaneously encouraging a fair distribution of labor (second term). This yields a best possible performance of one, while the worst performance would receive zero.

Currently, the simplifying assumptions in place disallow missed events, but in the future, a third term should be introduced to account for this as well.

While it is clear that the stated form does not allow all values in the range $[0, 1]$ (for example, a performance of one would require energy expenditure to be identically zero, which is physically impossible), it will nonetheless still provide a relative preferential ordering between different controllers.

3.2. Results With The Microscopic Model

For comparison, several different scenarios are considered. A summary of the various results can be found in Figure 6.

When nothing is known about the source, often algorithms must be adjusted to account for assumptions exploited in the periodic case. In fact, none of what we have done in the threshold-based controllers relied on assumptions about source behavior, and therefore running them unaltered in this new situation shows nearly identical performance. As mentioned previously, we consider both periodic (10s period, 10% duty cycle) and aperiodic (1s events, random arrival rate) event sources.

3.2.1. Always-On vs. Always-Off

As previously mentioned, a node which is active, with its radio on, through the entire day of simulation will use approximately $\hat{E}_{max} = 1.5$J (Figure 6, column 1). As this is basically the worst imaginable case for a control mechanism, we will consider this to be the upper bound on possible energy consumption, and use it for the value of the normalizing constant in equation 1, to ensure that the output values scale properly.

Likewise, a node who sleeps the length of the simulation will use only $\hat{E}_{min} = 38.0$mJ. Again, this case is unrealistic and undesirable, principally due to the fact that it results in a network incapable of accomplishing anything (neither sensing nor computation), but it will function as an adequate lower bound.

Obviously, neither of these situations are practical or even useful in a real environment, but they set boundaries with which we can compare our results, and understand where in the spectrum they fall.

3.2.2. Strict Baton-Passing

In the engineered solution (described in section 2.3) with a network of 9 nodes over 24 hours, if nothing goes wrong, average energy consumption is 428.5mJ, and the load is perfectly distributed (see Figure 6, column 2). These seem like respectable results, but we have already explained why they way in which they were obtained was overly optimistic.

3.2.3. Fixed Thresholds

Using the current stimulus randomization method (described in section 2.4), if thresholds are immutable and homogeneously distributed, then the result is the same as picking a node randomly to take over each new shift, as, when the stimulus is high enough to merit a response from any node, it will invoke responses from every node.

Heterogeneously distributed thresholds will enforce an order of preference, but as the hardware capabilities of the nodes in the network are designed to be the same (obviously, there will be minimal hardware differences, but it would be impractical if not impossible to attempt to assess these a priori for each node), there is no readily apparent way to

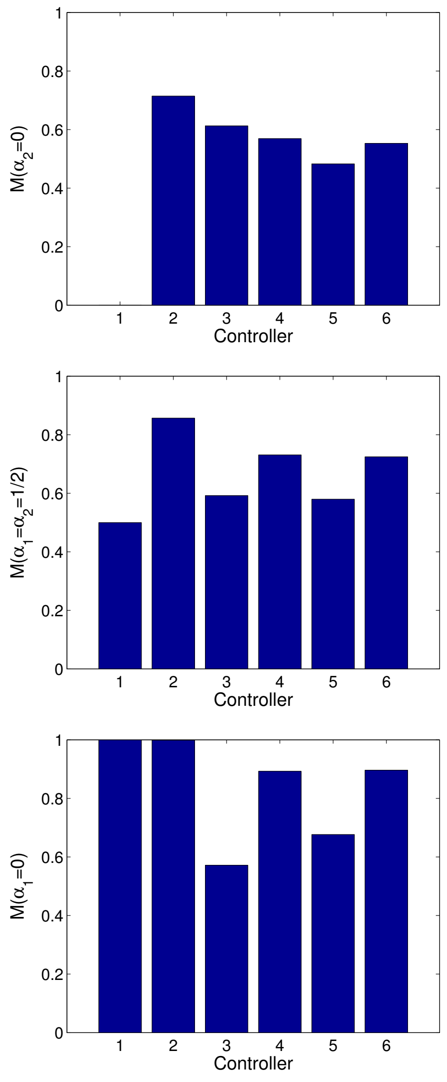

Figure 6. Network performance evaluation for different values of $\alpha_{\{1,2\}}$ (Equation 1). Top: $\alpha_{\{1,2\}} = \{1, 0\}$. Middle: $\alpha_{\{1,2\}} = \left\{\frac{1}{2}, \frac{1}{2}\right\}$. Bottom: $\alpha_{\{1,2\}} = \{0, 1\}$. Controllers: 1. Always On, 2. Engineered (periodic), 3. Fixed Thresholds (periodic), 4. Adaptive Thresholds (periodic), 5. Fixed Thresholds (aperiodic), 6. Adaptive Thresholds (aperiodic).

artificially assign heterogeneous thresholds in a meaningful manner.

Consequently, for ease of implementation, we have tested nodes with randomly distributed fixed thresholds. The result is as expected: the average power is higher, due to extended negotiation times when the initially generated stimulus is not large enough to provoke a response. There is also greater variation between nodes, as the ones with lower thresholds do more work (see Figure 6, columns 3 [periodic] and 5 [aperiodic]).

3.2.4. Adaptive Thresholds

To avoid exhausting the nodes with low thresholds, we can modify the threshold locally as a function of available remaining energy (battery level) at each shift change, according to the function:

$$\theta_{n+1} = \frac{1}{2}\left(\theta_n + \left(1 - B^3\right)\right) \qquad (3)$$

where B is the percentage of remaining battery life (see Figure 7). Our microscopic model incorporates a linearized model of battery depletion based on real measurements (Figure 4). According to simulations run, this appears to give more desirable results than a linear adjustment, but further investigation should be done in order to determine the optimal transformation.

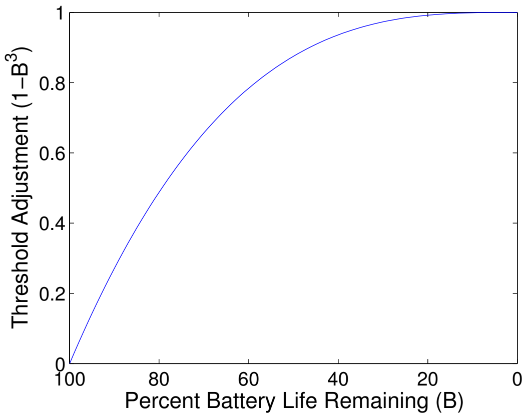

Figure 7. Threshold adjustment (second term in Equation 3). Various possibilities were attempted, including similarly shaped linear and exponential functions. The polynomial was chosen as a compromise between desirable behavior and computational simplicity (since it must be performed regularly on the node's limited microprocessor).

Adding this adaptation gives much better performance than the fixed case, and is closer to the engineered solution (see Figure 6, columns 4 [periodic] and 6 [aperiodic]).

3.3. Results in a Real Sensor Network

As shown in Figure 8 and Table 1, we implemented the same adaptive threshold-based controller in TinyOS and tested it on a small network of MICAz nodes. The minor differences between the simulated results and the hardware results can be explained by variation in initial battery voltages.

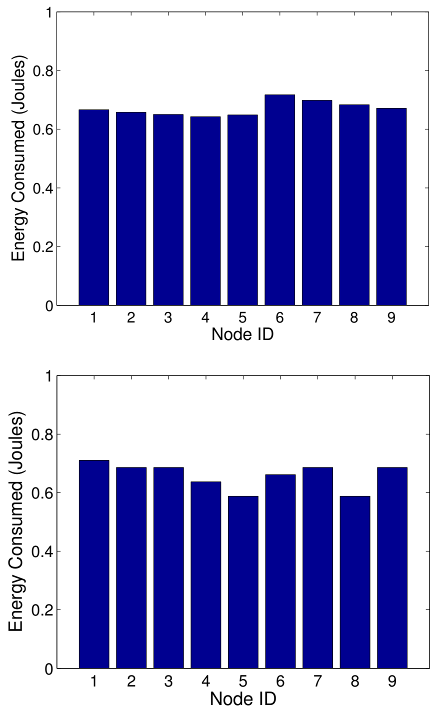

Figure 8. Power used by each node in a sensor network during 24 hours using energy-aware adaptive thresholds; in simulation (top), and on real hardware (bottom).

	$M(\alpha_1 = 1)$	$M(\alpha_1 = 1/2)$	$M(\alpha_1 = 0)$
micro model	0.5529	0.7246	0.8963
real network	0.6261	0.7130	0.8000

Table 1. Comparison of values predicted by the microscopic model with those observed in the physical network deployment.

4. CONCLUSION & FUTURE WORK

It has been demonstrated that threshold-based algorithms can be applied to the problem of energy-constrained load balancing in distributed wireless sensor networks in a potentially beneficial manner. In the future, this type of controller should be compared with results from other methodologies, such as market-based or game-theoretic algorithms. The authors also are also currently exploring possible extensions to this approach which will allow for maintaining an arbitrary number *k* nodes active at a time, and also networks with larger spacial distributions, without the assumption of complete connectivity.

5. ACKNOWLEDGMENTS

Christopher Cianci and Alcherio Martinoli are currently sponsored by a Swiss NSF grant (contract Nr. 11 P00268647/1). Vlad Trifa is currently sponsored by an EPFL undergraduate fellowship.

6. REFERENCES

[1] W. Agassounon and A. Martinoli, *Efficiency and Robustness of Threshold-Based Distributed Allocation Algorithms in Multi-Agent Systems* Proc. of the First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems AAMAS-02 (Bologna, 2002), ACM Press, 2000, pp. 1090–1097.

[2] W. Agassounon, A. Martinoli, and K. Easton, *Macroscopic Modeling of Aggregation Experiments Using Embodied Agents in Teams of Constant and Time-varying Sizes*, Autonomous Robots, Special Issue on Swarm Robotics **17** (2004), 163–192

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence From Natural to Artificial Systems*, Oxford University Press, New York, NY, 1999.

[4] R. R. Brooks and S. S. Iyengar, i *Maximizing Multi-Sensor System Dependability*, Proceedings of IEEE/SICE/RSJ 1996 Conference on Multisensor Fusion and Integration for Intelligent Systems, Washington D. C., IEEE, Piscataway, NJ, 1996, pp 1–7.

[5] H. Dai, R. Han, *A Node-Centric Load Balancing Algorithm For Wireless Sensor Networks*, IEEE Global Communications Conference (GLOBECOM), Wireless Communications, **1** (2003), 1-5, pp. 548–552.

[6] J. Hill and D. Culler, *Mica: A Wireless Platform for Deeply Embedded Networks*, IEEE Micro **22** (2002), 12–24

[7] M. Krieger and J.-B. Billeter, *The call of duty: Self-organized task allocation in a population of up to twelve mobile robots*, Robotics and Autonomous Systems **30** (2000), 65–84

[8] A. Martinoli, K. Easton, and W. Agassounon, *Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation*. Special Issue on Experimental Robotics, Int. Journal of Robotics Research, **23** (2004), No. 4, pp. 415–436.

[9] A. Martinoli, A. Ijspeert, and F. Mondada, *Understanding Collective Aggregation Mechanisms: From Probabilistic Modeling to Experiments with Real Robots.*, Special Issue on Distributed Autonomous Robotic Systems, Robotics and Autonomous Systems, **29** (1999), pp. 51–63.

[10] V. Shnayder, M. Hempstead, et al., *Simulating the Power Consumption of Large-Scale Sensor Network Applications*, Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems SenSys-04 (Baltimore, 2004), ACM Press, 2004