GEOFFREY MAINLAND AND
MATT WELSH

distributed, adaptive resource allocation for sensor networks

Geoffrey Mainland is currently a Ph.D. student at Harvard University and received his A.B. in Physics from Harvard College. His research interests include programming languages, distributed systems, networks, and intelligent control.

mainland@eecs.harvard.edu

Matt Welsh is an assistant professor of Computer science at Harvard University. His research interests encompass sensor networks, operating systems, and distributed systems.

mdw@eecs.harvard.edu

SENSOR NETWORKS HAVE THE POTEN-tial to revolutionize any number of fields. In the future, sensor networks may allow bridges to immediately report on structural damage following an earthquake, environmental monitoring systems to track pollutants in real time, and emergency response workers to direct limited resources to those who need medical attention most urgently. They will also vastly increase the quantity and quality of research data scientists can collect in the field. Our group at Harvard is working on several projects that apply sensor network technology to problems not traditionally associated with computer science, including CodeBlue, a software and hardware platform for emergency response, and the Volcán Tungurahua project, where we are using sensor networks to monitor eruptions at an active volcano in central Ecuador.

A typical sensor network device is the UC Berkeley Mica2 node, which consists of a 7.3MHz ATmega128L processor, 128KB of code memory, 4KB of data memory, and a Chipcon CC1000 radio capable of 38.4Kbps and an outdoor transmission range of approximately 300 meters. The node measures 5.7cm by 3.1cm by 1.8cm and is typically powered by two AA batteries, with an expected lifetime of days to months, depending on application duty cycle. With such limited computational and communication resources, how can effective applications be developed using this class of device? One might be tempted to assume that in a few years sensor network devices will have more powerful CPUs and better radios. Undoubtedly more powerful devices will appear, but we believe that the majority of sensor network nodes will be similar to the Mica2 in terms of processing power and bandwidth—they'll just be much smaller and cheaper. Far from becoming obsolete, techniques for programming these small devices will become increasingly important.

Consider two commonly cited applications for sensor networks: environmental monitoring [1, 2] and distributed vehicle tracking [3, 4]. Both applications require nodes to collect local sensor data and relay it to a central base station, typically using a multi-hop routing scheme. To reduce bandwidth requirements, nodes may need to aggregate their local sensor data with that of other nodes. Even these simple applications present unique challenges to system implementers. Nodes

must individually determine a schedule for sampling, aggregating, and sending data, subject to energy budget constraints. This schedule affects energy usage and, therefore, the overall lifetime of the network, as well as the quality of the data generated by the network. A node's ideal schedule is based on its physical location, position in the routing topology, and changes in the environment. As a result, there is almost never an ideal a priori common schedule for all nodes. Any action-scheduling algorithm must cope with network dynamics, so even full knowledge of a node's network location and capabilities doesn't allow one to choose a schedule ahead of time that will work well for all settings.

Many current applications use a single fixed, common schedule anyway, or try to build in some ad hoc adaptive behavior. For example, an application might selectively activate nodes that are expected to be near some phenomenon of interest. The only current tool that programmers have to address the scheduling issue is manual tuning, which is difficult and error-prone.

We propose an adaptive resource allocation scheme for sensor networks, called "Self-Organizing Resource Allocation" (SORA). Rather than defining a fixed node schedule, SORA causes nodes to individually tune their rate of operation using techniques from reinforcement learning [5]. Nodes receive rewards for taking "useful" actions that contribute to the overall network goal, such as listening for incoming radio messages or taking sensor readings. Each node learns which actions are profitable based on this reward feedback. Network retasking is accomplished by adjusting rewards, rather than pushing new code to sensor nodes, and network lifetime is controlled by constraining nodes to take actions that meet a local energy budget.

Application Example: Vehicle Tracking

As a concrete example of using SORA to manage resource allocation in a realistic sensor network application, we consider tracking a moving vehicle through a field of sensors. Vehicle tracking raises a number of interesting problems in terms of detection accuracy and latency, in-network aggregation, energy management, routing, node specialization, and adaptivity [4, 6, 7]. Vehicle tracking can be seen as a special case of the more general data collection problem also found in applications such as environmental and structural monitoring [2, 8].

In the tracking application, each sensor is equipped with a magnetometer capable of detecting local changes in a magnetic field, which indicates the proximity of the vehicle to the sensor node. One node acts as a fixed base station, which collects readings from the other sensor nodes and computes the approximate location of the vehicle based on the data it receives. The systemwide goal is to track the location of the moving vehicle as accurately as possible while simultaneously maximizing the efficiency of the network's energy use.

Each sensor node can take the following set of actions: *sample* a local sensor reading, *send* data toward the base station, *listen* for incoming radio messages, *sleep* for some interval, and *aggregate* multiple sensor readings into a single value. Each node maintains a fixed-length FIFO buffer of sensor readings, which may be sampled locally or received as a radio message from another node. Each entry in the buffer consists of a tuple containing a vehicle location estimate weighted by a magnetometer reading. The sample action appends a local reading to the buffer, and the listen action may add an entry if the node receives a message from another node during the listen interval.

Each action a has a utility $u(a)$ given by:

$$u(a) = \begin{cases} \beta_a r_a & \text{if the action is } \textit{available} \\ 0 & \text{otherwise} \end{cases}$$

where r_a is the current reward for action a, and β_a is the *estimated probability of payment* for that action, which is learned by nodes as described below. In our work, reward vectors are broadcast by the base station, but they may also be static parameters of the sensor network program. An action may be *unavailable* if either the current energy budget is too low to take the action, or other dependencies have not been met (such as lack of sensor readings to aggregate). This utility function is just the expected reward for taking a given action.

The estimated probability of receiving a reward for an action a, β_a, is updated every time that action is taken using an exponentially weighted moving average (EWMA). The equation for this update is:

$$\beta'_a = \begin{cases} \alpha + (1-\alpha)\beta_a & \text{if } a \text{ receives a reward} \\ (1-\alpha)\beta_a & \text{otherwise} \end{cases}$$

where α represents the sensitivity of the EWMA filter. If an action does not produce a reward, the node's estimated probability of receiving a reward decreases, but if the action does produce a reward, the estimated probability increases. Nodes learn the probability of receiving a reward instead of directly learning the expected reward for an action because this allows them to more easily adapt when a new reward vector is injected into the network.

A node chooses an action to perform by examining the utilities of all available actions and picking the action with the largest utility, which is just the expected reward. The expected reward for an action will vary over time due to possible reward adjustments and changing environmental conditions. Therefore, it is important that nodes periodically "take risks" by choosing actions that have a low reward probability β_a. To allow nodes to occasionally explore the action space, we employ an ϵ-greedy action selection policy. With a small probability, ϵ, when faced with a decision, a node will choose an available action uniformly at random. With probability 1- ϵ the node selects the "greedy" action, that is, the action that maximizes the utility $u(a)$. This exploration prevents a node from ever again selecting an action that has been unprofitable in the past.

Comparison with Existing Approaches

To compare the use of SORA with more traditional approaches to sensor network scheduling, we implemented three additional versions of the tracking system. The first employs static *scheduling,* in which every node uses a fixed schedule for sampling, aggregating, and transmitting data to the base station. Nodes perform a round of actions: sample, listen, aggregate, and transmit. The node then sleeps for a period of time. Given a daily energy budget, a node determines how long it must sleep between these rounds to meet this energy budget. The same schedule is used for every node in the network, so nodes do not learn which actions they should perform, nor do they adapt their sampling rate to environmental changes such as the approach of the vehicle. This is the typical approach used by current sensor network applications.

The second approach employs *dynamic scheduling,* in which nodes continuously adjust their sleep period based on their current remaining energy. This allows nodes that do not consume energy aggregating or transmitting data to use this conserved energy to increase their sampling rate.

The third and final approach, the *Hoods tracker,* is based on the tracking system implemented using the Hoods communication model [7]. It is largely similar to the dynamically scheduled tracker except in the way that nodes calculate the target location. Each node that detects the vehicle broadcasts its sensor reading

to its neighbors. The node then listens for some period of time and, if its own reading is the maximum of those it has heard, computes the centroid of the readings (based on the known locations of neighboring nodes) as the estimated target location. This location estimate is then routed toward the base station. We implemented the Hoods tracker to emulate the behavior of a previously published tracking system for direct comparison with the SORA approach.

For purposes of comparison, we are interested in two metrics: tracking accuracy and energy efficiency. We do not expect SORA to be more accurate than the other scheduling approaches, but if it is to be a realistic solution it should perform similarly. Our goal is to give good performance while maximizing energy efficiency, so we are willing to sacrifice some accuracy for more efficient energy usage.

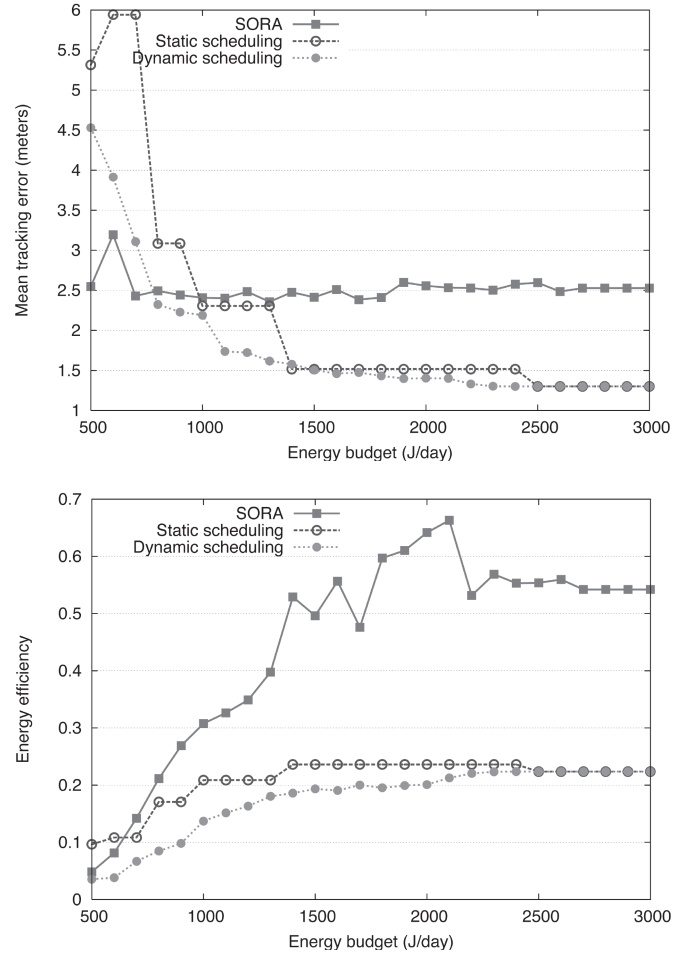

FIGURE 1: TRACKING ACCURACY AND ENERGY EFFICIENCY

Figure 1 summarizes the accuracy and efficiency of each scheduling technique as the energy budget is varied. Each system varies in terms of its overall tracking accuracy as well as in the amount of energy used. While SORA has a somewhat higher rate of tracking error compared to the other scheduling techniques, it demonstrates the highest efficiency, exceeding 66% for a daily energy budget of 2100J. The static and dynamic schedulers achieve an efficiency of only 22%. In SORA, most nodes use far less energy than the budget allows. The ability of SORA to "learn" the duty cycle on a per-node basis is a significant advantage for increasing network lifetimes. The Hood tracker performs poorly due to its different algorithm for collecting and aggregating sensor data, so it is not included in Figure 1.

Conclusions

Current approaches to resource management are often extremely low-level, requiring that the operation of individual sensor nodes be specified manually. With SORA, nodes self-schedule their local actions in response to feedback. This allows nodes to *automatically* adapt to changing conditions and specialize their behavior in response to physical location, routing topology, and environmental changes. While it does require a reformulation of application logic, using SORA is not particularly difficult, and it allows sensor networks to utilize resources much more efficiently than standard, ad hoc manual techniques. The increased complexity of future sensor network applications may make them a bad fit for this simple technique, but SORA is not meant to be universal. We view it not as a final solution, but as an important step that demonstrates the potential power of adaptive techniques in real sensor network systems.

REFERENCES

[1] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *Proceedings of the Workshop on Data Communications in Latin America and the Caribbean*, 2001.

[2] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson, "Wireless Sensor Networks for Habitat Monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02),* Atlanta, GA, USA, Sept. 28, 2002.

[3] Dan Li, Kerry Wong, Yu Hen Hu, and Akbar Sayeed,"Detection, Classification and Tracking of Targets in Distributed Sensor Networks," *Journal of the IEEE Signal Processing Magazine,* Vol. 19, No. 2, March 2002.

[4] Yingqi Xu and Wang-Chien Lee, "On Localized Prediction for Power Efficient Object Tracking in Sensor Networks," *Proceedings of the 1st International Workshop on Mobile Distributed Computing,* May 2003.

[5] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[6] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proceedings of the IEEE,* November 2003.

[7] Kamin Whitehouse, Cory Sharp, Eric Brewer, and David Culler, "Hood: A Neighborhood Abstraction for Sensor Networks," *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MOBISYS '04),* Boston, MA, June 2004.

[8] Venkata A. Kottapalli, Anne S. Kiremidjian, Jerome P. Lynch, Ed Carryer, Thomas W. Kenny, Kincho H. Law, and Ying Lei, "Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring," *Proceedings of the SPIE 10th Annual International Symposium on Smart Structures and Materials,* San Diego, CA, March 2000.