SpotON: Just-in-Time Active Event Detection on Energy Autonomous Sensing Systems

Yubo Luo
Department of Computer Science
UNC Chapel Hill
yubo@cs.unc.edu

Shahriar Nirjon
Department of Computer Science
UNC Chapel Hill
nirjon@cs.unc.edu

***Abstract*—We propose SpotON, which is an *active* event detection system that runs on harvested energy and adapts its sleeping cycle to match the distribution of the arrival of the events of interest. Existing energy harvesting systems wake up periodically at a fixed rate to sense and process the data to determine if the event of interest is happening. In contrast, SpotON employs reinforcement learning to learn the pattern of events at run-time and uses that knowledge to wake itself up when events are most likely to happen. Being able to remain asleep more often than a fixed wake-up system, SpotON is able to reduce energy waste, increase the amount of harvested energy, and be able to remain active for longer period in time when the events of interest are more likely to occur. We conduct a simulation-driven experiment to compare our proposed solution with a fixed-schedule system and results show that SpotON is able to capture 2–5X times more events and is 3–12X more energy-efficient than the baseline.**

***Index Terms*—Energy harvesting, Q-learning, Event detection.**

I. Introduction

As the development of computational power, computing algorithm and hardware, more and more stand-alone and sustainable applications are emerging, pushing the world forward to the ultimate instantiation of the Internet of Things (IoT) – the “Smart Dust”. Smart Dust is a system of many tiny microelectromechanical systems that are energy autonomous [3]. However, current IoT world is dominated by battery-powered systems which are bulky and unsustainable. Energy harvesting is one of the ways that can lead us to the final instantiation of IoT.

Energy harvesting systems harvest energy from various energy sources, such as RF, piezoelectric or solar. It eliminates the need for replacing batteries and enables energy autonomy which is crucial to long-term sensing applications.

To achieve energy autonomy, we have to overcome challenges resulting from energy harvesting. First, computation in energy harvesting systems is intermittent and this causes problems because most computing algorithms are long-running programs. There is literature addressing how to enable correct execution of existing long-term running programs on energy harvesting systems by guaranteeing atomicity, data consistency, forward progress [7], [8]. Second, energy supply in energy harvesting systems is so limited that energy efficiency is crucial. Dewdrop [4] takes iterative tasks as a scheduling problem and dynamically changes the starting voltage based on the size of the next task to improve energy efficiency. Capybara [8] deploys an array of capacitors with different capacitance and dynamically changes the capacitance of the system. Mayfly [10] considers the timeliness of data and discards stale data to avoid wasting energy in learning outdated data. Third, event detection is a typical and important task in IoT world, but active event detection has yet not been well studied. Active event detection means that the event itself can not produce a trigger signal to wake up the micro-controller (MCU), and it requires the MCU to actively sense the event. More detailed explanation about passive and active event detection is described in Section II. There is one approach called Monjolo [5] addressing passive event detection in energy harvesting systems. It uses the energy harvested from the event itself as a trigger to wake up the system and thus avoids active sensing. However, Monjolo only applies to cases where the event itself can generate a certain amount of energy that Monjolo uses as a trigger. As far as we know, there is no literature addressing cases where the target event itself can not serve as a trigger. Thus, we propose SpotON, the first energy harvesting system that deals with active event detection by waking up just in time.

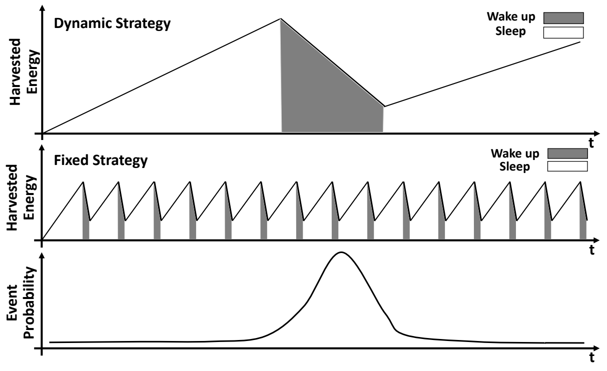


Fig. 1. Comparison between fixed and dynamic strategies. The fixed strategy can only detect a small fraction of total events but the dynamic strategy only wakes up during event-active time intervals and thus detects more events.

Most existing intermittently powered systems have predefined turn-on and turn-off thresholds, which means the system periodically wakes up at each capacitor charging cycle. Though some of them dynamically change the thresholds according to the complexity of the next task, they are still in the category of periodically charging and discharging, as shown in Figure 1. The key to enabling active event detection in energy autonomous systems is to ensure that there is enough energy left in the storage capacitor to power up the microcontroller when the event is about to happen. If the system magically knows when the event is about to happen, it can wake up more frequently during this time interval but remain powered-off more frequently at other times. SpotON learns the event pattern, predicts when the event is more likely to happen, keeps the system in powered-off more frequently when events are not active, and saves surplus energy in a dedicated capacitor array which can compensate the massive energy consumption of frequent waking-ups when events are active. In this way, we can borrow energy harvested from previous charging cycles.

In real-life scenarios, events behaves in a pattern which may change over time. SpotON uses a reinforcement learning algorithm – Q-learning to learn the event pattern online. If the event pattern changes over time, SpotON learns the new pattern and updates the system, making itself a "real-time" event detector.

II. Problem

The key to enabling active event detection is to decide when to wake up the MCU. One important part of the waking up process is the trigger mechanism. We classify the waking up trigger into the following three categories [6]: periodic, opportunistic and event-based.

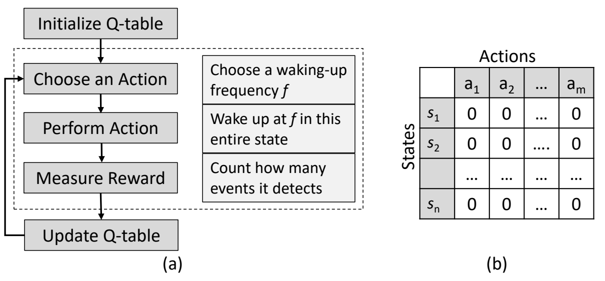
The periodic trigger wakes up the MCU at a fixed period, which requires a predictable energy supply. This type of trigger is rare in intermittent systems. The opportunistic trigger is common in intermittent systems which depends on the harvestable energy and a predefined voltage threshold. If the voltage of the storage capacitor reaches V_{th}, the trigger is activated, and the MCU keeps running until the voltage decreases to V_{min}. This type of trigger usually requires a dedicated voltage detection circuit and a power management module [8]. The event-based trigger wakes up the MCU based on an expected event. The system needs to harvest enough energy before a trigger event happens in order to be woken up successfully. If an event happens before enough energy is stored up the system will miss this event. The detection rate of this type of trigger is bounded by the capacitor recharging rate [5], [6]. SpotON can be considered as an event-based trigger but it applies to entirely different applications. The event-based trigger mentioned in [5], [6] only applies to cases where the event itself can actively activate a trigger signal, and this signal is leveraged to wake up the MCU. This trigger-activating process does not involve MCU. Possible applications for this type of event-based trigger include door opening detection [6] where the action of opening the door itself can vibrate a piezo sensor and generate a trigger signal and airflow monitoring [11] where the airflow itself can also vibrate a piezo sensor and generate a trigger signal. This type of sensing is passive. The MCU just passively waits for an waking-up signal.

However, there are many applications whose target event can only be passively captured by active sensing controlled by the MCU, e.g., wildlife watering monitoring and environment noise detection. This type of applications requires active sensing from the MCU. SpotON is specially designed for this type of event detection. It learns the event pattern and uses it as an internal event-based trigger. SpotON only benefits cases where the event happens in a certain pattern that we can use some learning algorithm to learn. If the target event happens randomly without any pattern, then it is not a proper candidate application for SpotON.

We call our internally triggered waking-up approach a dynamic strategy, compared to the commonly used approaches which are static or fixed. Current fixed systems use up all energy immediately once they are charged to a threshold voltage and wake up. They do not borrow energy from previous charging cycles. By saying SpotON a dynamic strategy, we do not mean that the threshold voltages or the storage capacitance is dynamic, we mean that after the system has already harvested enough energy to wake up, it dynamically chooses to wake up at which frequency level, e.g., high frequency, low frequency or even not waking up, based on the learned event pattern. In this way, SpotON is able to wake up just in time.

III. Design Overview/ Implementation

To wake up the system in a proper time at a proper frequency based on the event pattern, we use reinforcement learning to implant this intelligence into SpotON.


Initialize Q-table
Choose an Action
Perform Action
Measure Reward
Update Q-table
Choose a waking-up frequency f
Wake up at f in this entire state
Count how many events it detects
(a)
Actions
States
(b)

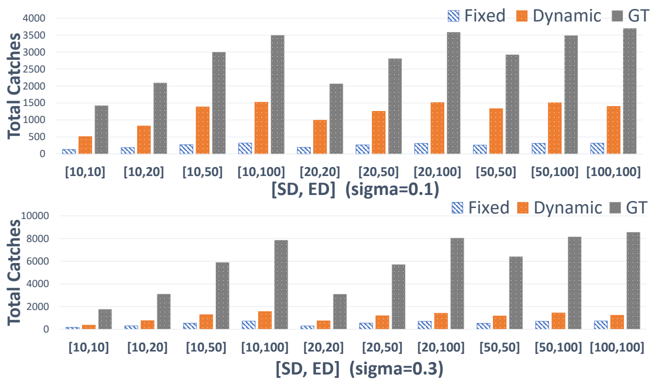
	a_1	a_2	...	a_m
s_1	0	0	...	0
s_2	0	0	....	0
	...	...	...	...
s_n	0	0	...	0

Fig. 2. (a) Workflow of Q-learning [1]; (b) Initialized Q-table.

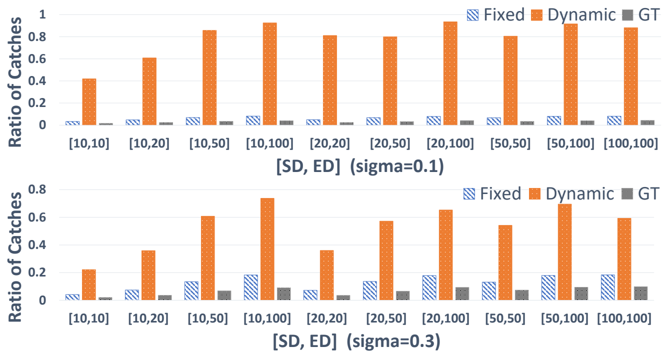
A. Q-Learning

Q-learning is employed to help SpotON learn the event pattern and make waking-up decisions. Q-learning [9] is an effective way of making optimal decisions to achieve the best reward based on past observations. The workflow of Q-learning is described in Figure 2(a).

The learned experience is stored in a Q-table which contains the weight for each state-action pair. We use $S = \{s_1, s_2, ..., s_n\}$ to denote the set of states and $A = \{a_1, a_2, ..., a_m\}$ the set of actions. In our case, one state s_i means a specific time interval. Adjacent states are adjacent time intervals. For example, if the time duration of each state is one hour and s_i is 9-10am, then s_{i+1} means 10-11am and s_{i-1} means 8-9am. Each state has the same duration length. One action a_j means a specific waking-up frequency. It could be



(a) Total catches



(b) The ratio of total catches to total waking-up time units

Fig. 3. Simulation results. There are two simulation settings. One contains 100 events sampled from $[\mu = 12, \sigma = 0.1]$ and 100 events sampled from $[\mu = 15, \sigma = 0.1]$. The other contains 100 events sampled from $[\mu = 12, \sigma = 0.3]$ and 100 events sampled from $[\mu = 15, \sigma = 0.3]$. SD is state duration and ED is event duration. The time unit is one second.

waking up MCU once every minute, or once every 10 minutes or once every hour, etc. Both the state duration and the waking-up frequency for each action are application-specific.

Q-learning is a table-based reinforcement learning algorithm. Q-table contains the weight of each state-action pair and these weights are updated at each step by calculating the reward gained from each step. One step corresponds to one state. The update equation of Q-table is as follows:

$$Q_{ij} = Q_{ij} + \alpha[R_{ij} + \gamma \max_{k=1,..,m}(Q_{i+1,k}) - Q_{ij}] \qquad (1)$$

where Q_{ij} denotes the weight of the state-action pair of (s_i, a_j); R_{ij} denotes the reward gained in state s_i by taking action a_j; α denotes learning rate; γ denotes discount rate; max calculates the maximum expected future reward given the new state and all possible actions at that new state.

Figure 2(b) shows the initialized Q-table with zeros which can also be initialized by training on an off-line dataset. Training a Q-table using off-line dataset before deployment saves the time spent in learning the event pattern from scratch.

B. Online Adaptive Ability

In a real life scenario, the event pattern is usually not fixed, and it may change as time goes by. For example, in the application of wildlife watering monitoring, wildlife in the desert comes to water site to drink water, and our system wants to take as many pictures containing target wildlife as possible. However, wildlife's watering habit varies in different seasons. The system must be capable of online learning to keep the Q-table updated to the real-time environment.

In Q-learning, there is a parameter called ϵ related to action-taking decisions. At the beginning of each step, a random number r is generated and compared to ϵ. If $r > \epsilon$ the system takes an action based on Q-table. Otherwise, a random action is taken. The value of ϵ keeps decreasing to a small number as Q-table is more and more well-trained. Once ϵ decreases to its minimum value, the system is well-trained, and the Q-table stops updating. Larger ϵ value means the system explores more and smaller ϵ value means the system exploits more. To implant online learning ability to SpotON we need to reset this ϵ regularly. As for how often we should reset ϵ, it depends on the application.

IV. Evaluation

We have conducted computer simulations to evaluate the performance of SpotON. Our simulations are based on a python package called *gym* which is a toolkit for developing and comparing reinforcement learning algorithms [2]. The most important parameters are as follows:

- state duration (SD): how long one state lasts;
- event duration (ED): how long one event lasts;
- energy bank capacity (EBCap): how much energy the storage capacitor can save at maximum;
- event distribution parameters (μ, σ): we assume the event obeys normal distribution. μ and σ are the mean value and the standard deviation respectively.

To simplify the simulation, we assume our system harvests solar power, the time unit is one second and the total simulation time is one day. The energy consumption rate is 10 times as the energy storage rate, which means 10-second charging can support 1-second discharging. Solar energy is only harvestable during the daytime, e.g. from 6 am to 8 pm. We set the EBCap to 2-hour continuously charging, which means the capacitor can at maximum store energy harvested from two hours.

If the system wakes up at one time unit and there is an event happening, then we consider this time unit a catch or a positive waking-up. The number of catches or positive waking-ups is one of our evaluation metrics, and it means how many time units the system wakes up when there is an event. The ratio of positive waking-ups to total waking-ups is our second metric. The reason why we use these two evaluation metrics is that in real life scenarios we care more about if the MCU wakes up at the event-happening moment, which is related to the efficiency of energy usage. Taking the wildlife watering as an example, the system wakes up and takes a picture of the water site. The more pictures that capture an animal, the more efficiently the system uses the harvested energy. Energy used for waking up without capturing an event is wasted. There is a negative reward for waking up without a catch and a positive reward for waking up with a catch.

We compare SpotON (**Dynamic**) with the other two systems, the ground truth (**GT**) and the fixed system (**Fixed**). The ground truth system is powered by a battery so it can catch all time units when there is an event, and its total waking-ups equal to the length of the simulation which is 24 hours in our case. The Fixed system is the existing energy harvesting system which wakes up after the capacitor is fully charged and uses up all energy immediately.

Figure 3(a) demonstrates that our SpotON has more catches than the fixed system. Especially in the case of $\sigma = 0.1$, SpotON has around 5 times as many event catches as the fixed system. Compared to the ground truth, there are still a lot of missed time units where there are events. But remember our goal is to make as many positive waking-ups as possible, rather than capturing all event-happening time units. We also notice that SpotON has better performance for event distribution with smaller standard deviation. The reason is that smaller standard deviation means the data are more converged and it is easier for Q-learning to learn the event pattern.

TABLE I
AVERAGED RESULT FROM ALL SIMULATION RESULTS

	Number of catches		The ratio of positive waking-ups	
	$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$
Fixed	256	515	6.5%	13.2%
Dynamic	1225	1114	79.6%	53.4%
GT	2855	5848	3.3%	6.8%

Figure 3(b) shows the ratio of positive waking-ups to total waking-ups. SpotON significantly outperforms the other two systems. The highest ratio is 0.92 which means in that case 92% of SpotON's waking-ups happen when there is an event. The fixed system has only 10% positive waking-ups on average which means 90% of harvested energy is wasted in useless waking-ups. As expected, the ground truth system has the lowest energy usage efficiency because it is powered up all the time. Table I shows the averaged result from all results.

V. Future Plan

Our future work includes tuning the parameters of the Q-learning algorithm, considering the time, energy, and complexity of event detection algorithms, and implementation of the system on a real hardware.

- Algorithm parameters: In the preliminary experiments, we find that there are still opportunities for improvements. In some settings, there are unused energy in the capacitor at the end of the experiment. Currently, we use the same reward parameters for all simulations. Tuning the reward parameter in a more fine-grained way can make the system use up all harvested energy and thus yield even better results.
- Generalization: currently, we assume that the events are easily detected. For example, if we consider loud noises as events of interest, we only need to use an audio sensor and compare the signal amplitude to a threshold. If it is higher than the threshold, we consider it as an event. However, to generalize SpotON to other applications, we have to consider the energy consumption due to executing the recognition algorithm, e.g. image-based event detection where image recognition is needed to figure out if a picture contains an object or an event we are interested in and this consumes much more energy than merely sensing audio signals.
- Hardware: one crucial prerequisite for SpotON is that we must have a large and dynamic storage capacitor. Let's look at the following example: we want to monitor wildlife watering, and we assume that the wildlife of interest come to the water site mostly in the afternoon. Our system gradually learns to wake up more frequently in the afternoon and sleeps more in the morning. The energy harvested in the morning must be stored somewhere and then be used to wake up MCU more frequently in the afternoon. Thus, saving such energy would require a variant design of Capybara [8] that can dynamically change the storage capacitance. This requires a complete energy management unit facilitated by specialized hardware and software design.

In summary, SpotON opens a new way for energy autonomous systems to detect events actively by waking up just in time and by adapting its internal model of the environment as the real-world environment changes. We also plan to implement SpotON on different energy harvesting platforms.

References

[1] Diving deeper into reinforcement learning with q-learning. https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe. Accessed: 2019-02-20.

[2] Gym toolkit. https://gym.openai.com/. Accessed: 2019-02-20.

[3] Smart dust - wikipedia. https://en.wikipedia.org/wiki/Smartdust. Accessed: 2019-02-20.

[4] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: an energy-aware runtime for computational rfid. In *Proc. USENIX NSDI*, pages 197–210, 2011.

[5] B. Campbell, M. Clark, S. DeBruin, B. Ghena, N. Jackson, Y.-S. Kuo, and P. Dutta. perpetual sensing for the built environment. *IEEE Pervasive Computing*, 15(4):45–55, 2016.

[6] B. Campbell and P. Dutta. An energy-harvesting sensor architecture and toolkit for building monitoring and event detection. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 100–109. ACM, 2014.

[7] A. Colin and B. Lucia. Chain: tasks and channels for reliable intermittent programs. *ACM SIGPLAN Notices*, 51(10):514–530, 2016.

[8] A. Colin, E. Ruppel, and B. Lucia. A reconfigurable energy storage architecture for energy-harvesting devices. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 767–781. ACM, 2018.

[9] M. E. Harmon and S. S. Harmon. Reinforcement learning: A tutorial. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1997.

[10] J. Hester, K. Storer, and J. Sorber. Timely execution on intermittently powered batteryless sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, page 17. ACM, 2017.

[11] T. Xiang, Z. Chi, F. Li, J. Luo, L. Tang, L. Zhao, and Y. Yang. Powering indoor sensing with airflows: a trinity of energy harvesting, synchronous duty-cycling, and sensing. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 16. ACM, 2013.