Sensing Together: Cooperative Task Adaptation and Scheduling for IoT-Nets using Renewable Energy

Elizabeth Liri[*‡], K. K. Ramakrishnan[*], Koushik Kar[†], and Flavio Esposito[‡]
[*]Dept. of Computer Science and Engineering, University of California Riverside, Riverside, CA, USA
[†]Electrical, Computer, and Systems Engineering Dept., Rensselaer Polytechnic Institute, Troy, NY, USA
[‡]Computer Science Dept., Saint Louis University, St. Louis, MO, USA

***Abstract*—IoT devices used in various applications, such as monitoring agricultural soil moisture, or urban air quality assessment, are typically battery-operated and energy-constrained. We develop a lightweight and distributed cooperative sensing scheme that provides energy-efficient sensing of an area by reducing spatio-temporal overlaps in the coverage using a multi-sensor IoT network. Our "Sensing Together" solution includes two algorithms: Distributed Task Adaptation (DTA) and Distributed Block Scheduler (DBS), which coordinate the sensing operations of the IoT network through information shared using a distributed "token passing" protocol. DTA adapts the sensing rates from their "raw" values (optimized for each IoT device independently) to minimize spatial redundancy in coverage, while ensuring that a desired coverage threshold is met at all points in the covered area. DBS then schedules task execution times across all IoT devices in a distributed manner to minimize temporal overlap. On-device evaluation shows a small token size and execution times of less than 0.6s on average while simulations show average energy savings of 5% per IoT device under various weather conditions. Moreover, when devices had more significant coverage overlaps, energy savings exceeded 30% thanks to cooperative sensing. In simulations of larger networks, energy savings range on average between 3.34% and 38.53%, depending on weather conditions. Our solutions consistently demonstrate near-optimal performance under various scenarios, showcasing their capability to efficiently reduce temporal overlap during sensing task scheduling.**

***Index Terms*—multi-sensor IoT; distributed scheduler; energy efficiency; task adaptation; cooperative sensing; comb placement problem**

I. Introduction

IoT devices are used in various application areas, such as agriculture and urban monitoring. They are typically resource-constrained with limited battery, memory, and computation. One technique for energy management includes using renewable energy sources like solar combined with solar prediction mechanisms to maximize device lifetime. Micro-local conditions at each device like shade/foliage affect its received solar energy, limiting the accuracy and effectiveness of a centralized energy management solution. Therefore, decentralized, lightweight IoT energy management solutions that are adaptive to local device conditions are highly desirable

Multi-sensor IoT devices can monitor different phenomena, e.g., in agriculture, IoT devices may have temperature sensors and cameras, but this means more energy used per IoT device. However, if multiple IoT devices cover the same geographical area, cooperative sensing ensures more energy-efficient operations. Cooperative sensing allows multiple IoT devices to coordinate their sensing operations, reducing the duplication of sensing tasks and temporal overlap between neighboring devices. While comprehensive coverage depends on the number of sensors, energy efficiency remains paramount. Saving energy reduces operational costs and environmental impact, while enhancing the system longevity and reliability. Inter-device communication costs are high, so distributed and cooperative sensing solutions should minimize this cost in addition to being adaptive and independent of deployment patterns.

There are a number of examples of IoT energy management solutions that include renewable energy, such as Signpost [1], FarmBeats [2], and SEMA [3]. However, these do not leverage cooperation between IoT devices to use energy judiciously and perform sensing without duplication. Cooperative sensing is a scheduling problem, and has also been used for energy management by avoiding duplication of sensing. For example, [4] uses a partially observable Markov decision process for target tracking, [5] uses cooperative monitoring with mobile applications and [6] uses a frequency scaling power minimization approach for scheduling. However, these solutions are complex [4], aimed at less constrained devices and do not account for limited renewable energy sources [5], or require additional hardware [6]. Our distributed solution is simple to implement and is designed to operate efficiently on energy-constrained IoT devices by minimizing inter-device communication.

Our goal is to provide an on-device energy management solution that addresses the constraints of limited renewable energy sources and achieves energy savings from cooperative sensing between multiple IoT devices. Our distributed solution eliminates the need for a centralized controller to schedule each IoT device and we aim to optimize energy utilization while maximizing sensing efficacy.

Our "Sensing Together" solution presented here needs initial sensing rates for each sensor at each IoT device. These can be provided and optimized at each IoT device by existing solutions such as SEMA-A [3]. Through distributed coordination, our Sensing Together cooperative sensing solution saves energy by adapting the sensing rates to reduce spatial and temporal overlap using two algorithms: *Distributed Task Adaptation (DTA)* and *Distributed Block Scheduler (DBS).*

We thank the US NSF for their generous support through grants CNS-1818971 and CPS 2133407.

DTA adapts the sensing rates of each sensor in an IoT device to minimize redundancy in spatial coverage, subject to meeting a minimum coverage threshold at all points in the monitored area. Adaptation is done through distributed coordination using a distributed token passing protocol run over a Distributed Hash Table (DHT) structure, using two traversals in opposite directions (forward and reverse) to balance energy used in all devices. We theoretically show that DTA meets the coverage threshold at all points, as long as the IoT devices in the network have enough energy to achieve it.

DBS minimizes temporal overlap by moving the sensing patterns around in time. We reduce the sensing task scheduling question to a novel block scheduling problem and present an algorithm for minimizing temporal overlap. We theoretically show that the algorithm finds the optimal solution to the block scheduling problem in $O(\Delta^2)$ time, where Δ is related to the token size. The token size determines the degree of coordination between devices, and captures the balance of energy savings versus computation and communication complexity.

Our on-device experiments show execution times of less than 0.6s per device on average. In small network simulations using experimentally measured values, DTA achieves energy savings of 5% per IoT device on average; however, in areas with more redundancy due to overlapping coverage areas, energy savings for some devices exceeded 30%. In large networks, our results show an average energy savings of 3.34% on cloudy days and up to 38.53% on sunny days. Higher energy availability on sunny days means more frequent sensing resulting in higher redundancy across neighboring devices; DTA reduces this redundant sensing. Furthermore, in various scenarios, DBS consistently demonstrated performance close to the optimal solution, showcasing its capability to efficiently reduce temporal overlap during scheduling.

II. Related Work

Examples of IoT energy management solutions include SEMA [3], Signpost [1], and FarmBeats [2]. Each solution incorporates a renewable energy component, IoT device sensing operations, and an energy management system. SEMA [3] offers an on-device energy management system utilizing an approximate Model Predictive Control approach to optimize information utility and sensing operations while minimizing energy consumption. Signpost [1], designed for urban monitoring, features a solar prediction component leveraging historical data, modular sensors, and a sensor virtual battery management system. FarmBeats [2], aimed at farm monitoring, incorporates a weather-aware solar prediction mechanism and conserves energy through duty-cycling of base station components. SEMA's solar prediction approach considers local device conditions without relying on weather forecasts like FarmBeats. Signpost employs a uniform solar prediction for all devices in a given area. In terms of energy management, the mechanisms used by SEMA and Signpost operate at the device level, whereas FarmBeats manages energy at the base station level. Furthermore, SEMA accounts for the specific energy needs of individual IoT applications, while Signpost assumes uniform energy requirements for all applications. However, none of these solutions utilize cooperative sensing to harness information from neighboring devices, a fundamental feature of the solution we present here.

The solutions in [7]–[10] use a task scheduling approach to manage energy at the IoT device. Authors in [7] use a dynamic programming approach for scheduling. [10] uses a Mixed Integer Linear Programming (MILP) formulation to define the sensing problem. Jarvsis [9] uses a hierarchy of control tasks operating in the Cloud/Fog. LSA [8] determines whether all task deadlines can be met before scheduling.The solution in [10] runs on a centralized node and not on the IoT device, and both [10] and [7] are complex solutions. The communication cost in [9] can be significant, and individual IoT devices cannot be scheduled independently. Moreover, none of these solutions take advantage of cooperative sensing. Our algorithms are designed to be simpler, employ tokens for inter-device communication, and reduce communication costs by sharing minimal information between neighbors.

Cooperative sensing solutions for energy management have been proposed in [4]–[6], [11]–[13], and a comprehensive survey on cooperative sensing in IoT networks is in [14]. For scheduling, [4] used the partially observable Markov decision process while [11] used a randomized myopic policy, selecting devices with the highest energy levels to perform sensing in each time slot. Authors in [6] use a power minimization approach where nodes receive task requests with estimated task execution times and schedule the tasks by scaling the CPU core's operating frequency ensuring task completion within the estimated time. In [12] a near-optimal transmission scheduling policy to maximize throughput using cooperative sensing is presented while [5] uses cooperative monitoring with mobile applications. Our previous work [13] minimizes overlapping sensing times between IoT devices by only sharing free time slots. Information is shared using tokens and devices independently schedule their start times. [6] requires additional hardware for frequency scaling, [4] is complex, and [5] targets mobile phones which may not be as strictly energy-constrained and also have more compute capability. These limitations hinder the effectiveness of their solutions on more energy-constrained IoT devices. While [13] is lightweight and distributed, it ignores potential overlaps in sensing times later in the sensing period and thus fails to fully exploit the benefits of cooperative sensing.

The algorithms we develop in this paper are hardware-independent, less complex, and optimized for efficient operation on energy-constrained IoT devices while accounting for any available renewable energy. They are lightweight, distributed, and take full advantage of cooperative sensing by considering all sensing operations within the sensing period.

III. System Design

A task refers to a distinctive sensing modality used by an IoT device's sensor e.g., a temperature task for a device with a temperature sensor. Each task is characterized by a task parameter such as number of measurements/sensing operations

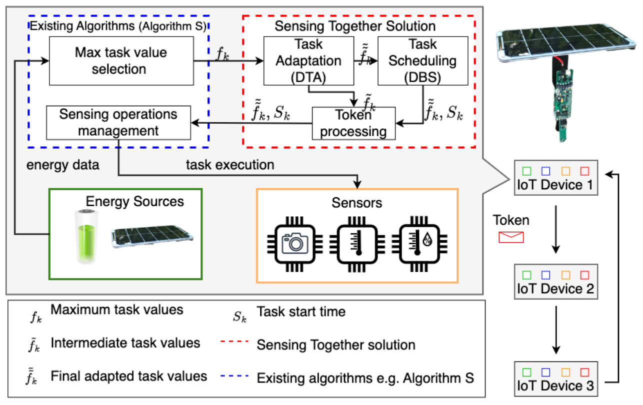

Existing Algorithms (Algorithm S)
Max task value selection
f_k
Sensing Together Solution
Task Adaptation (DTA)
$\tilde{\tilde{f}}_k$
Task Scheduling (DBS)
$\tilde{f}_k$
$\tilde{\tilde{f}}_k, S_k$
Token processing
$\tilde{\tilde{f}}_k, S_k$
Sensing operations management
energy data
task execution
Energy Sources
Sensors
IoT Device 1
Token
IoT Device 2
IoT Device 3
f_k Maximum task values
S_k Task start time
$\tilde{f}_k$ Intermediate task values
Sensing Together solution
$\tilde{\tilde{f}}_k$ Final adapted task values
Existing algorithms e.g. Algorithm S

Fig. 1: Sensing Together: Architecture Overview.

per epoch (15 min period). Task parameters have minimum and maximum values e.g. the maximum temperature task parameter value is 15. Existing energy and task management solutions can determine the maximum task parameter values possible at each IoT device independently based on its available device energy. However, geographical and temporal overlap can cause redundant data and unnecessary transmissions, wasting energy at the device. Cooperative monitoring allows IoT devices to share information and coordinate sensing to address these challenges. Our goal is to develop a lightweight and efficient distributed cooperative monitoring solution that minimizes inter-device communication overhead. We show our "Sensing Together" solution architecture in Fig. 1. We rely on any existing solution to provide the initial task parameter values and have used the SEMA-A algorithm from [3] in this work. We propose two on-device distributed algorithms that reduce spatio-temporal overlap in two token rounds using our token passing protocol: A Distributed Task Adaptation (DTA) algorithm (§ IV) that adapts the task values based on neighbors task information and a Distributed Block Scheduler (DBS) (§ V) that schedules task sensing operations. Next, we review SEMA-A in [3] (§ III-A), describe the abstractions forming the basis of our algorithms in § III-B, and discuss the communication setup in § III-C.

A. *Decentralized Task Value Computation and IoT Hardware*

There are three main differences in the SEMA-A implementation used in this work for decentralized task value computation. First, in [3] image quality was the image task parameter; however we fixed the image quality, and used number of images taken per epoch n_i (ranging from 1–30). Second, instead of the night algorithm, we always use the SEMA-A algorithm for task selection. Third, we only consider image, temperature, and humidity tasks. The prototype IoT device used in [3] is based on a Raspberry Pi Zero W, supports WiFi and has multiple sensors including a camera and temperature and humidity sensors (see Fig. 1). It is powered by a Li-ion battery and solar panel.

1) Task Energy and Utility Characterization: The utility function in SEMA-A is a simple concave utility function of the form $U^k = 1 - e^{-f^k/x}$, where f^k is the task parameter for task k, and x is a task-dependent utility constant.The task energy models were determined by executing multiple experiments and fitting a curve. The total energy used per device is the sum of the base (idle) energy for device operations and additional energy for each task E^k characterized by its variable task parameter f^k. For the image, temperature, and humidity tasks, the equation for the additional energy used has the form, $E_k = f^k(E^{k,base} + E_u s^{k,base})$, where $k = i, t$ or h for image, temperature, and humidity; $s^{k,base}$ is the size of the sensor data uploaded, and E_u (Joules/MB) [15] is the TCP upload transmission energy per MB. For image $E^{i,base}$= 0.80943J, $s^{i,base}$=3.516MB, (see [3] for more details).

2) Model Predictive Control (MPC) Formulation: [3] uses a Model Predictive-based approach for IoT energy management that caters to local environmental conditions at each IoT device. The goal is to choose task parameters that maximize the sensing utility over a time horizon of M epochs (till 06:00am the next day), subject to battery charging characteristics/limits and renewable energy supply predictions. This is given by the MPC formulation in Eqns. 1-5, where task k has a sensing optimization variable f^k with a minimum value f^k_{min}, a utility function U_k, and an assigned weight w_k. E_{min} and E_{max} are the minimum and maximum battery levels allowed. $R(m)$ represents the energy pushed into or taken out of the battery during epoch m and has a maximum value of $R_{max}(m)$ i.e. recharge limit of the battery. $E_k(f^k(m))$ represents the total energy required by task k in epoch m when its task value is set to f^k. The predicted solar energy available during epoch m is $S(m)$ (given by the solar prediction mechanism proposed in [3]), the base energy required by the device while idle is E_{base} and $\hat{S} = S(m) - E_{base}$. The maximum battery voltage is v_{max}, epoch duration in seconds is e (we use 15 min epochs from [3]) and the current flowing into the battery is I. Here, all tasks have equal weight.

$$\max_{f^k(m) \geq f^k_{min}} \sum_{m=m_0}^{M} \sum_{k \in \mathcal{K}} U_k(f^k(m)), \tag{1}$$

subject to:

$$E(m) = \min\left[E(m-1) + R(m), E_{max}\right], \ \forall m, \tag{2}$$

$$R(m) = \min\left[\hat{S}(m) - \sum_{k \in \mathcal{K}} E_k(f^k(m)), R_{max}(m)\right], \forall m, \tag{3}$$

$$R_{max}(m) = \frac{IeE(m)v_{max}}{E_{max}}, \ \forall m, \tag{4}$$

$$E(m) \geq E_{min}, \ \forall m. \tag{5}$$

The MPC formulation represents a convex optimization problem, but without any specific structure that would allow it to be solved efficiently within the small computing and memory capabilities of a typical IoT device. However, under a reasonable approximation that the charging limit $R_{max}(m)$ does not vary appreciably with the battery charge level, an assumption that is experimentally justified in [3], an algorithm, SEMA-A, that solves the MPC in low time complexity is proposed. SEMA-A rewrites Eqns. 1- 5 transforming the optimization problem into an energy allocation problem (Eqn. 6) and a sensor scheduling problem (Eqn. 7) where $g(m) = \sum_k E_k(f^k(m))$.

$$\max_{g\in G, g(m)\ge g_{min}} \sum_{m=m_0}^{M} V(g(m)), \qquad (6)$$

$$\text{where } V(g(m)) = \max_{g(m), f^k(m)\ge f^{k,min}} \sum_{k\in\mathcal{K}} U_k(f^k(m)). \qquad (7)$$

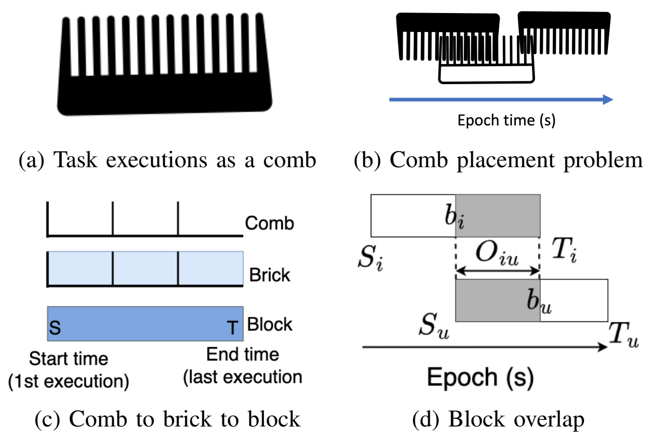
SEMA-A uses this bi-level decomposition and a novel combination of a recursive dynamic programming-like procedure and an incremental max-min fair allocation method to solve the approximate MPC problem in cubic time complexity ($O(M^3 + MK^2)$), where K is the total number of tasks and M is the total number of epochs. While the details of the SEMA-A algorithm are not necessary to understand the contribution of this paper, it is worth noting that SEMA-A runs *independently* on each IoT device and provides the initial task values (which are also the upper bounds). Using these task values (without adaptation) and scheduling sensing times without any coordination would result in high spatial and temporal coverage overlap which we address with our DTA and DBS algorithms.

B. Spatial and Temporal Coverage Abstraction

Let IoT device i have an ID s_i and f_i^k represent task k's value computed by the device independently (using the SEMA-A algorithm). We adapt the f_i^k value cooperatively through token-based coordination using DTA and then schedule the sensing operations using DBS. Since our cooperative sensing objective is to minimize the sensing overlap, the spatial and temporal coverage model has to be chosen accordingly.

Spatial Overlap Abstraction: Geographical coverage overlap determines which devices are neighbors and their overlapping coverage amount. Two IoT devices are *coverage neighbors* (or simply *neighbors*) if they have non-zero overlap in their coverage areas. If we discretize the area of interest into a set of grid points, two IoT devices are neighbors if their coverage areas share at least one grid point. While discretization is unnecessary for our framework, we use it for ease of exposition. Based on task type, coverage area shapes may differ; e.g., temperature and image tasks have circular and conical shapes respectively. Devices may thus have different sets of neighbors per task. The goal of DTA (Section IV) is to reduce the amount of overlap coverage between all neighbors that sense the area in an epoch, across all the grid points.

Temporal Overlap Abstraction: If we view sensing (an instantaneous operation for each task) at an IoT device as a spike on a timeline, then the scheduling question involves finding where to place the spikes so the sensing instances per grid point are spaced out to provide the best temporal coverage of the grid point. To reduce computation and communication complexity for coordination, we assume that every epoch the f_i^k spikes at device i are uniformly spaced in time, with an interval $w_i^k = e/f_i^k$ between them (e is the epoch duration). This lets us view the sensing process for a task at an IoT device as a comb (Fig. 2a). Thus, with multiple IoT devices (multiple combs) running the same task, scheduling all task executions to maximize temporal coverage becomes equivalent to a *comb placement problem*, where the combs are placed to maximize the inter-spike gap at each grid point. (see Fig. 2b). The optimal comb placement problem is still quite complex to formulate and solve since each comb (device) has a different number of spikes (determined by DTA) and therefore different inter-spike times. Towards developing a low-complexity solution, we "spread out" the spike from one execution to the next, forming a "brick". The spreading out of the spike is reasonable because for example in periodic sensor measurements, data obtained from one measurement is considered valid/fresh until the next periodic measurement is due. Therefore multiple spikes transition into multiple bricks placed side by side, since the task executions are uniformly distributed within the epoch. The multiple bricks side by side can then be represented by a single *block* characterized by a device ID s_i, a start time S_i^k when the first brick starts, an end time T_i^k when the last brick ends, and a block height $1/w_i^k$ where w_i^k is the task period representing the number of seconds between successive task executions (sensing actions). Tasks with higher number of task executions have shorter inter-task intervals, and hence higher block height. Fig. 2c shows this comb-to-brick-to-block task abstraction.



(a) Task executions as a comb (b) Comb placement problem (c) Comb to brick to block (d) Block overlap

Fig. 2: Task Abstractions

The task scheduling problem is now framed as a block placement problem where we only consider temporal overlap with neighboring nodes. Therefore at a given node, our goal is to place the current node's block on top of its neighbor's blocks to minimize total overlap with the blocks of all its neighbors aggregated across all commonly covered grid points.

C. Communication Protocol Overview

Our algorithm does not depend on deployment, but requires devices to communicate with neighbors (i.e., nodes with overlapping coverage) using the underlying wireless network.

Inter-device communication: We assume the devices communicate with their neighbors using a logical, circular Distributed Hash Tables (DHT) network [16]. Communication between neighbors in the DHT may be either through an access point or a relay (if using device-to-device communication). Therefore, we assume the DHT can be constructed such that the peering relationship reflects the coverage neighbor relationship as described in Section III-B. Each IoT device

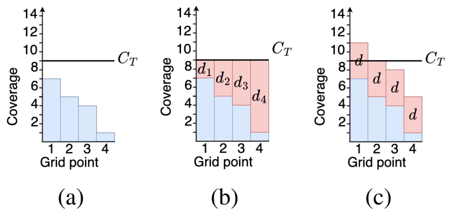


Fig. 3: Token 1 (N-S direction)

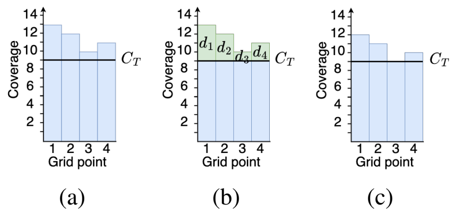


Fig. 4: Token 2 Case 1 (S-N direction)

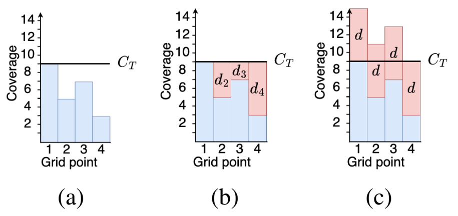


Fig. 5: Token 2 Case 2 (S-N direction)

knows the address of its next live neighbor and is resilient to connectivity changes as is typical with DHTs. Connectivity among neighbors is maintained independently of our scheduler. WiFi is used in [3], however, our protocol is independent of the communication technology. Other technologies like LoRA, BLE, etc., can be used for extra energy savings.

Token Passing: We use token-passing to share information between IoT devices and assume that existing token ring techniques (IEEE 802.5) [17] can address challenges like handling lost or duplicate tokens and leader election. The token carries information from upstream nodes (IoT devices that have already run our algorithms). The upstream nodes' information carried in the token includes s_i and task information including S_i^k, w_i and f_i^k. We assume the token is large enough to carry sufficient geographical neighbor information for all upstream neighbors of the current device.

IV. Task Adaptation

SEMA-A determines the upper bound task parameter values for each device. We introduce the term "coverage threshold" (C_T) representing the minimum number of measurements (sensor readings) required per grid point, per epoch, and per task. For instance, a temperature task with $C_T = 9$, requires a minimum of 9 temperature measurements over each covered grid point per epoch. The application determines the value of C_T which only applies to grid points covered by the IoT network. The measurements to achieve C_T can be provided by a single IoT device or multiple devices working together. This requires coordination between neighboring devices, when multiple devices cover the same grid points.

The coverage value, denoted by c_g^k, is the current total number of measurements for task k in an epoch over grid point g. It is determined by adding the total number of task executions (i.e., f_i^k values for all i devices) for task k from all devices covering grid point g in an epoch. For example, if we have 3 IoT devices (s_1, s_2, and s_3) covering grid point $g = 10$ and they have maximum temperature task values 5, 6, and 2 respectively (where $k = t$ for the temperature task). If $C_T = 9$ then one solution to meet this threshold is to use task values 4, 3, and 2 at s_1, s_2, and s_3 respectively. This means that during an epoch s_1 takes 4 measurements, s_2 takes 3 measurements and s_3 takes 2 measurements resulting in total grid point coverage of grid point 10, $c_{10}^t = 9$. The mechanism by which this is done is discussed later in this section.

If $c_g^k > C_T$ (*over coverage*), energy is wasted because the extra measurements are not required. If $c_g^k < C_T$ (*under coverage*), the grid point is under-covered and insufficient sensing information is being provided. Our target is to ensure that for all covered grid points every epoch $c_g^k = C_T$ (*threshold coverage*), which satisfies the coverage requirements without wasting energy. Using the neighbor's task information, DTA adapts the task values at an IoT device to achieve this. Tokens traverse the network in two rounds and share information between IoT devices.

A. Forward Traversal of Token (Token 1)

The token travels say in a North-South direction in the network, as determined by the communication architecture (Section III-C). Consider an example where the token travels through devices in ascending ID order. IoT device 5 (s_5) covers grid points 1-4 which are also covered by other IoT devices 1-4. Figs. 3a-3c illustrate the algorithm's operation for task k when the token arrives at s_5. DTA extracts the neighbor's task k information from the token, saves it, and determines the current coverage of all its grid points. This is shown by the blue bars in Fig.3a, with coverage values $c_1^k = 7, c_2^k = 5, c_3^k = 4$ and $c_{g4}^k = 1$ for grid points 1-4.

DTA finds the difference between the coverage values and C_T for each grid point, (shown by the red bars $d_1 - d_4$ in Fig.3b). DTA then finds the mean of the differences, i.e. $d = mean(d_1, d_2, d_3, d_4) = mean(2, 4, 5, 8) = 4.75 \approx 4$, and the updated *intermediate* task value for s_5 $\tilde{f}_5^k$ is the minimum of the upper bound task value f_5^k from SEMA-A and d i.e. $\tilde{f}_5^k = min(f_5^k, d)$. Fig. 3c shows the updated coverage values with the new $\tilde{f}_5^k$ for s_5 included. Here we assumed $\tilde{f}_5^k = d$ and therefore the updated coverage values are $c_1^k = 7 + d = 11, c_2^k = 5+d = 9, c_3^k = 4+d = 8$ and $c_{g4}^k = 1+d = 5$ for grid points 1-4 respectively. The token is updated with s_5 data and forwarded to the next node. This distributed approach allows each device to adapt its values based only on information from its upstream neighbors.

B. Reverse Traversal of Token (Token 2)

The second token travels in the reverse direction carrying information from devices that were "downstream" when using token 1 to nodes that were previously "upstream". Once the token arrives at a device, it uses the "upstream" neighbor information saved when processing token 1, the "downstream" neighbor information carried in token 2, and its own intermediate task value $\tilde{f}_i^k$ to calculate the current grid point coverage. In our example, information from IoT devices 6, 7, and 8 (which were previously downstream) is now carried to IoT device s_5 which combines this information with the previously received information from IoT devices 1-4. The device now has information from all its neighbors.

Case 1: All grid points coverage is greater than C_T. This is an over-coverage case illustrated by the blue bars in Fig. 4a, which represent the current coverage per grid point and exceed the coverage threshold C_T. Note that these coverage values are calculated including the intermediate task value $\tilde{f}_5^k$ at s_5. DTA finds the excess coverage per grid point represented by $d_1 = 4$, $d_2 = 3$, $d_3 = 1$ and $d_4 = 2$ and shown by the green bars in Fig. 4b. DTA then finds the maximum excess coverage value that can be removed while ensuring all coverage remains at least C_T (d_3 in this case). The final task value is the intermediate value determined in the forward round minus the maximum excess coverage that can be deleted i.e. $\tilde{\tilde{f}}_5^k = \tilde{f}_5^k - d_3$, reducing over-coverage. Fig. 4c shows the final grid point coverage with the updated $\tilde{\tilde{f}}_5^k$ values. In this case since $d_3 = 1$ the updated coverage values are $c_1^k = 13 - d_3 = 12, c_2^k = 12 - d_3 = 11, c_3^k = 10 - 1 = 9$ and $c_4^k = 11 - d_3 = 10$ for grid points 1-4 respectively.

Case 2: At least one grid points coverage is less than C_T. This is an under-coverage case shown by the blue bars in Fig. 5a. DTA finds the under coverage per grid point ($d_2 = 4$, $d_3 = 2$, and $d_4 = 6$ shown by the red bars in Fig. 5b). DTA then finds the maximum under-coverage difference (d_4 here) and tries to minimize this. Therefore the final task value is the minimum of the upper bound (from SEMA-A) and the intermediate task value plus the maximum under-coverage difference, i.e. $\tilde{\tilde{f}}_5^k = min(f_5^k, \tilde{f}_5^k + d_4)$, which reduces under-coverage. Assuming the final task value $\tilde{\tilde{f}}_5^k = \tilde{f}_5^k + d_4$ the updated coverage values are then $c_1^k = 9 + 6 = 15, c_2^k = 5 + 6 = 11, c_3^k = 7 + 6 = 13$ and $c_4^k = 3 + 6 = 9$ for grid points 1-4 respectively. Note that although we do have over-coverage because the final task value $\tilde{\tilde{f}}_5^k$ affects coverage of all grid points covered by s_i, we prioritize ensuring we meet the coverage threshold over reducing the over-coverage.

C. Analysis

In the token forward traversal, each device only ensures an average coverage level of C_T over all the grid points in its coverage area; individual grid points can still be short of the required coverage threshold C_T. This deficit is made up during the token reverse traversal. The following result argues that, if the network has enough energy, each grid point will be covered up to C_T at the end of the reversal traversal step.

Theorem 1. *The desired coverage level (C_T) for task k at grid point g is satisfied at the end of the reverse traversal of the token, as long as $\sum_{i \in N_g} f_i^k \geq C_T$.*

Proof. Let N_g be the nodes in the DHT covering grid point g. Also, let $\hat{f}_g^k = \sum_{i \in N_g} \tilde{\tilde{f}}_i^k$ be the total coverage of grid point g for task k at the end of the reverse traversal step. Since each node $i \in N_g$ schedules up to rate f_i^k to meet the threshold C_T in the reverse traversal step, the only way the threshold C_T may not be met is if $\sum_{i \in N_g} f_i^k < C_T$, which contradicts the assumption. This proves the result. □

Theorem 1 is more of a *feasibility* result showing our solution attains threshold C_T if it is possible to attain it with the task parameters given by the SA algorithm. One goal of task adaptation is to *avoid coverage redundancy*. This depends on the level of coordination, determined by the number of devices whose task parameter information is carried by the token, which we denote by Δ. Extreme values, $\Delta = 0$ represents no coordination while $\Delta = N_{DHT}$ (number of DHT devices) represents full coordination among all the DHT devices. Under uniformity assumptions on the device deployment pattern and DHT topology, it can be argued that the amount of redundancy (over the required C_T) goes down with Δ as $O(1/\Delta)$. On the other hand, the message length grows as $O(\Delta)$, capturing the trade-off between efficiency and message complexity.

V. SCHEDULING

Intuitively, the sensing times of neighboring IoT devices should be *staggered* as much as possible to minimize the possibility that those devices are all sensing the same area simultaneously. For complexity reasons, we use uniformly distributed sensing times for the K sensors (tasks) at a device (see Section III-C), with frequency given by SEMA-A and further adapted by DTA (see Section IV). In the block sensing model (Section III-B), the only variable to optimize is the start time of the sensing block subject to fitting the entire block in an epoch. Here, we describe how to do it optimally (under the protocol constraints) and in a computationally efficient manner.

A. Block Scheduling Model

For each IoT device sensing pattern, represented as a block (Section III-B), our goal is to minimize the temporal coverage overlaps by multiple sensors covering the same area. Since the sensors are scheduled sequentially according to their DHT position, when a sensor is scheduled, our objective is to place its sensing block to minimize coverage overlap with all prior sensing blocks whose positions have already been chosen. The sensor block is scheduled in the DTA reverse traversal step, once the task value is finalized. Therefore, when the token reaches device i, it carries information on the starting time S_u^k and task frequency information $\tilde{\tilde{f}}_u^k$ for Δ sticks traversed immediately before stick i in the DHT. Since the sensing points $\tilde{\tilde{f}}_i^k$ and their inter-sensing interval w_i^k are known, the width and height of the sensing block are known for each task k. Therefore, only the start time S_i^k (and end-time T_i^k) needs to be optimized, given the start times of the Δ devices in the DHT traversed immediately before device i, to minimize their spatial overlaps in coverage with i being scheduled. For device u, let α_{iu} denote its spatial coverage overlap with i.

Let $O_{iu}(S_i^k)$ denote the overlap of the two devices when the start time of task k sensing block for i (currently being scheduled) is S_i^k (variable), while the position of the sensing block of u is fixed (pre-determined) (see Fig. 2d). The overlap is calculated by considering the shaded area formed by overlaying the sensing blocks of i and u, and counting when they

Algorithm 1 Distributed Block Scheduler Algorithm

```
Require: $token; b_i = [s_i, S_{i,dline}, T_{i,dline}, f_k = \frac{1}{w_i^k}]$
  $output \leftarrow []$
  $blocks \leftarrow createDeltaBlocks(token)$
  $wall \leftarrow createWall(blocks)$
  $y \leftarrow getTransitionPoints(wall)$
  for $j = 0 .. len(y)$ do
    $S_1 \leftarrow y[j]; T_2 \leftarrow y[j]; S_2 \leftarrow T_2 - w_i^k f_k$ {Place block $b_i$ start and end boundaries at transition point}
    for $S\ in\ [S1, S2]$ do
      if $0 \le S \le S_{deadline}$ then
        $b_i \leftarrow updateStartTime(S)$
        $overlap \leftarrow calcOverlap(b_i, blocks)$ {Eqn.9}
        $output.append([S, overlap])$
  $S_i^k \leftarrow getMinOverlap(output)$
```

overlap. The optimization goal is to choose S_i^k that minimizes overlap with all Δ devices scheduled prior to i, i.e.,

$$S_i^{*k} = \arg\min_{S_i^k} \sum_u \alpha_{iu} O_{iu}(S_i^k), \quad \text{where} \tag{8}$$

$$O_{iu} = \left(\frac{1}{w_i^k} + \frac{1}{w_u^k}\right)\left[\min(T_i^k, T_u^k) - \max(S_i^k, S_u^k)\right]_+; \tag{9}$$

$$T_i^k = S_i^k + w_i^k * f_i^k. \tag{10}$$

Here, α_{iu} represents the geographical coverage overlap between devices i and u (0 indicates no overlap at all) and $[z]_+ = \max(0, z)$. In general, the function $O_{iu}(S_i^k)$ is neither convex nor concave. This implies that standard convex optimization tools cannot solve this problem. Further, even though S_i^k is a scalar, it varies over a continuous space, making it a one-dimensional continuous optimization problem. However, we can utilize the linearity of the problem to develop an algorithm that computes the optimum block start time S_i^{*k} in (8)-(10) in time complexity $O(\Delta^2)$, described below.

B. Distributed Block Scheduling Algorithm

The DBS algorithm (Algorithm 1) describes scheduling of a single task, and is run separately for each task at a device. To explain the operation of DBS, we use the example of device s_5 ($i = 5$) scheduling task k. Device s_5 has four already scheduled neighbors (s_1-s_4) with their information in the token. When device i (5) receives the token, for each task k it extracts the information (S_j^k, w_j^k) for each of the Δ (4 in this example) device j included in the token. Device i generates blocks $b_1 - b_4$ for the prior Δ devices (Fig. 6b), with respect to which it must place its own block (see Fig. 6a) optimally.

The optimal placement of device i's block can be visualized as follows. The Block Scheduler arranges the prior Δ blocks along a timeline to create a "wall structure" where blocks with overlapping time spans are placed one on top of another (see Fig 6c). The figure assumes that the weights α_{iu} are the same for all prior blocks u, else they would have to be scaled accordingly. From the figure, it is clear that time points where the wall height changes $(t_i, t_2, \cdots)$ – or "transition points" (Fig 6d) – correspond to the start or end points S_u^k, T_u^k of all the prior blocks u. Clearly, there are up to 2Δ such transition points. As the block of device i slides along this time axis, the total value of the overlap function changes only when the block's start time S_i^k or end time T_i^k, hits a transition point. The steps where $S_5^k = t_1, S_{5,dline}$ and $T_5^k = t_3, t_4$ are illustrated in Figs.7a-7d

Theorem 2. *There exists an optimum solution S_i^{*k} such that either S_i^{*k} or T_i^{*k} is a transition point, i.e., either S_i^{*k} or T_i^{*k} equals S_u^k or T_u^k for some prior block u.*

Proof. For the sake of contradiction, assume that all optimum solutions of (8)-(10) are such that neither S_i^{*k} nor T_i^{*k} are transition points. Consider any such optimum solution.

Consider any prior block u among the Δ prior blocks. From (9), we see that increasing S_i^k "slightly" (from S_i^{*k}) would increase O_{iu} linearly with slope β_{iu}, where β_{iu} is either 0, $(1/w_i^k + 1/w_u^k)$, or $-(1/w_i^k + 1/w_u^k)$, until either S_i^k or T_i^k reaches a transition point. This continues until this increase in S_i^k is large enough (say $\delta_u^+ > 0$) for either S_i^k or T_i^k to become a transition point. Since O_{iu} is linear, decreasing S_i^k "slightly" (from S_i^{*k}) would then naturally increase O_{iu} linearly with slope $-\beta_{iu}$; this continues until the decrease in S_i^k is large enough (say $\delta_u^- > 0$) for either S_i^k or T_i^k to become a transition point. Let $\beta = \sum_u \alpha_{iu}\beta_u$.

First consider the case $\beta > 0$. In this case, we increase S_i^k by $\delta^+ = \min_u \delta_u^+$, which results in either S_i^k or T_i^k becoming a transition point, while the objective function in (8), $\sum_u \alpha_{iu}O_{iu}(S_i^k)$ strictly increases (from $\sum_u \alpha_{iu}O_{iu}(S_i^{*k})$). This contradicts the assumption that S_i^{*k} is optimal.

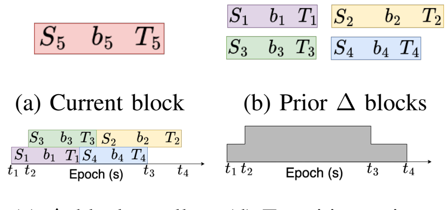
For the case $\beta < 0$, the argument is similar. In this case, we decrease S_i^k by $\delta^- = \min_u \delta_u^-$, resulting in either S_i^k or T_i^k becoming a transition point, while the objective function in (8), $\sum_u \alpha_{iu}O_{iu}(S_i^k)$ strictly increases from $\sum_u \alpha_{iu}O_{iu}(S_i^{*k})$. This again contradicts the assumption that S_i^{*k} is optimal.

Finally, for the case $\beta = 0$, we either increase S_i^k by δ^+ or decrease it by δ^-, which keeps the objective function the same at $\sum_u \alpha_{iu}O_{iu}(S_i^{*k})$ (optimum), but results in either S_i^k or T_i^k becoming a transition point. This also contradicts our assumption that in all optimum solutions, neither S_i^{*k} nor T_i^{*k} are transition points. Since all the three cases lead to a contradiction, this proves the result. ☐

Theorem 2 allows us to find S_i^{*k} by only computing the overlap function (8) corresponding to the 4Δ transition points. Each computation of the overlap function takes $O(\Delta)$ time, which implies that the optimum starting time of block i can be found in $O(\Delta^2)$ running time. This method is illustrated in Figure 7, and at a high-level in Algorithm 1.

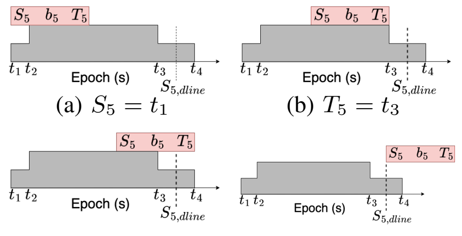
VI. Evaluation and Results

We use on-device and simulation experiments to evaluate the performance of our algorithms. For the simulations, we estimated energy use and coverage based on measured device data over several days (under different weather conditions). We consider a small network as a base case and then study a larger deployment. The evaluation metrics are total experiment



(a) Current block (b) Prior Δ blocks (c) Δ blocks wall (d) Transition points

Fig. 6: Construction of the wall for blocks



(a) $S_5 = t_1$ (b) $T_5 = t_3$ (c) $T_5 = t_4$ (d) $S_5 = S_{5,dline}$

Fig. 7: Placing block b_5 at transition points

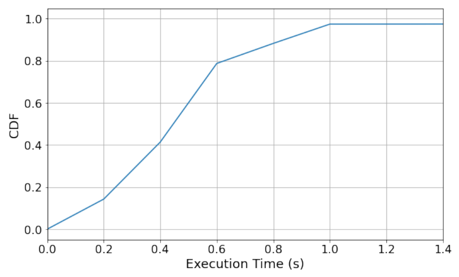
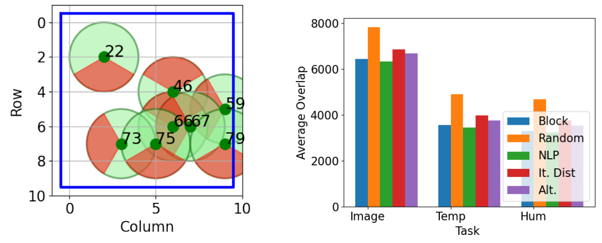


Fig. 8: DTA and DBS Execution time

TABLE I: Token and tasks epoch communication energy

	Min Energy(J)	Data(B)	Max Energy (J)	Data(B)
Image	1.318998	$3.516x10^6$	39.569936	$105.48x10^6$
Temp	0.000103	275	0.001547	4,125
Hum	$2.063x10^{-5}$	55	0.000309	825
Token	0.000178	474	0.000263	700



(a) Non-uniform deployment (b) Average overlap

Fig. 9: Baseline results without task adaptation

energy used and average total temporal overlap. We compared the performance of "Sensing Together" against several schedulers described next.

A Random Scheduler (Random) that selects a random start time from 0 to the task start deadline, as the token travels to each device. This ensures all task executions are completed within the epoch. A Centralized Optimal Scheduler (NLP) that operates at the sink, and uses data from all devices. A Gurobi optimizer solves the Non-linear Program which is the best case bound. The optimization problem for a single task k is given by $\min \sum_{pq} \alpha_{pq}^k \max\left(0, O_{pq}^k\right)$ for any two devices s_p and s_q. It minimizes the overlap between all IoT devices in the network while ensuring the overlap between any two IoT devices p and q (which is calculated using Eqns. 9 and 10) is always positive. An Iterative Distributed Scheduler (It. Dist.) that repeatedly randomly schedules an IoT device using DBS until the solution converges. It also uses *all* of the neighbor's information, not just upstream neighbors like DBS, without placing any token size restrictions. An Alternate Scheduler (Alt.) which alternately schedules the task start times to be 0 or the start deadline (limit) as the token travels between devices.

A. On-device Experiments

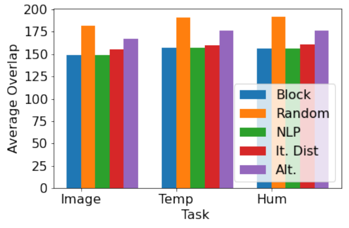
We deployed our Sensing Together solution onto three devices. The initial token size is 474 Bytes and increases by approximately 113 Bytes each time another device processes it. An increase in the token size results in a slight increase in the total execution time for the DTA and DBS algorithms. However, 80% of the time it is less than 0.6s. Fig. 8 shows the cumulative distribution on all the devices over several days.

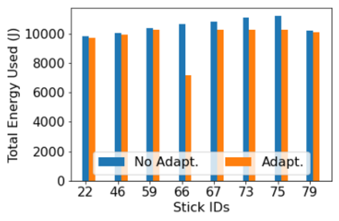
The communication energy for the tasks and token was calculated using the equations described in Section III-A1. Table I shows that the maximum token communication energy (in this 3-node network) is comparable to the lowest energy task (humidity) operating at maximum frequency. The last node in the DHT transmits the largest token and so uses the maximum token communication energy. The energy for transmitting the token from the current device can be calculated from $0.000178 + (\Delta) * 4.2391 * 10^{-5}$. Here, 0.000178J is the minimum token transmission energy, $4.2391 * 10^{-5}$J is the additional energy required every time we add 133B of data to the token. These results show that our algorithm design coupled with a small token size gives a feasible solution enabling operation directly on the devices with minimal execution time and communication energy.

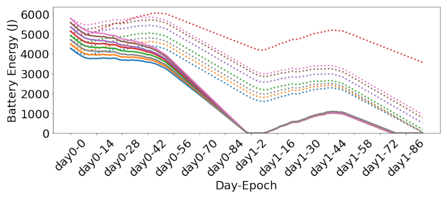
B. Simulation Experiments - Baseline Results

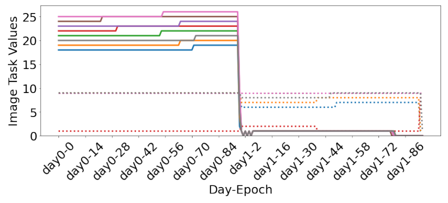
As a baseline experiment, we use a small network of 8 IoT devices deployed in a 10m x 10m grid. Each IoT device's ID corresponds to its grid point location and the IoT device IDs are 22, 46, 59, 66, 67, 73, 75, and 79. Each IoT device runs three tasks (image, temperature, and humidity sensing) with the variable parameter being frequency of execution. We use coverage threshold value $C_T = 9$ for all tasks. The coverage area radius is 2m for all tasks (see green circles and red cones in Fig. 9a). The experiment duration was 48 hours and we show results for a cloudy day solar profile for both days. We slightly varied solar patterns and the starting battery energy (which was approximated at 6,000J) for each IoT device. We use a non-uniform deployment (Fig. 9a) where some devices may have overlapping coverage (e.g., device 66). We first run the experiment with the DTA algorithm disabled and compare the performance of the five different schedulers. DBS results in the graphs are marked as "Block".

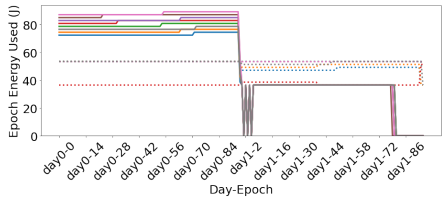
1) Schedulers **without** *DTA Adaptation:* Without DTA, all IoT devices use the initial task values from SEMA-A. The average temporal overlap of the schedulers (averaged across multiple experiment runs) is in Fig. 9b. Temporal overlap, measured in seconds, indicates the total time during


(a) Average overlap


(b) Total energy


(c) Battery energy


(d) Image task values


(e) Epoch energy

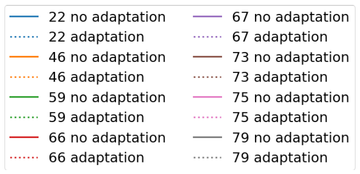
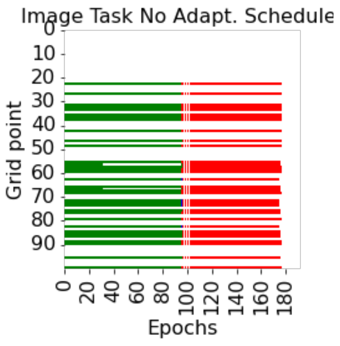
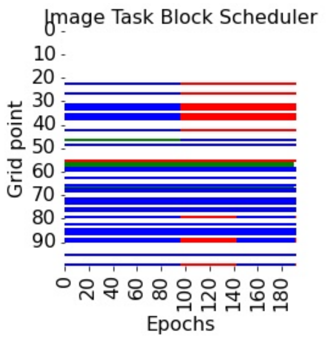

(f) Legend for Figs. 10c-10e

Fig. 10: Baseline Experiments with Task Adaptation (DTA)


(a) Image task (no adaptation)


(b) Image task (with adaptation)

(c) Legend

Fig. 11: Image grid point coverage with/without adaptation

an experiment when any two neighboring IoT devices are simultaneously performing sensing operations. Lower values are preferable. Neighboring IoT devices are those sharing spatial coverage of at least one grid point. The minimum and maximum temporal overlap values are determined by the task values at each device. DBS performs best with a low overlap value, close to the optimal (NLP) solution. The Iterative Distributed and Alternate schedulers perform slightly worse, while the Random scheduler has much worse performance. The Iterative Distributed Scheduler exhibits slightly poorer performance compared to DBS because its final converged overlap value is influenced by the initial "seed" chosen during simulation. Therefore, we averaged the results across multiple Iterative Scheduler iterations to get the final converged value which yields higher average overlap values. The solid lines in Fig. 10c show the starting battery energy levels per device and battery discharge pattern. The battery slowly discharges throughout day 1 for all devices till they all shut down temporarily early on day 2. However, high task values are still used for all tasks on day 1 (see Fig. 10d for the image task, no adaptation case). As more solar energy becomes available on day 2 even on this cloudy day, the IoT devices can charge their batteries, restart and perform sensing tasks. But on this cloudy day, the batteries cannot be sufficiently charged, and therefore only minimum task values are used. Between epochs 79-81 on day 2, all the IoT devices shut down again and do not recover (again, until the next day).

2) Benefit of Task Adaptation: Fig. 10a shows the average overlap for all tasks when DTA is added to the schedulers. Now, DBS performs best with values close to the optimal. The Random Scheduler performs the worst. Fig. 10b compares the total energy used per device using DBS and with (orange) and without (blue) DTA. Adaptation provides energy savings at all devices enabling them to operate over the 2 days (dashed lines in Fig. 10c show the battery changes at each device (legend in Fig. 10f)). In particular, device 66 has much higher energy saving due to its overlap i.e., all its grid points are also covered by its neighboring devices (see Fig. 9a).

Cooperative sensing with adaptation allows a device with overlaps e.g. device 66 to conserve energy by using min. task values, and thus DBS+DTA allows all tasks to be run over 2 days. Fig. 10d compares the initial image task values with no adaptation from SEMA-A (solid lines) and the final adapted task values (dotted lines) from DTA (similar results were seen with the other tasks). On day 2 when some neighbors use lower task values due to lower energy availability e.g., devices 46 and 67 between epochs 1 and 71, DTA at device 66 increases its task value. Thus, ensuring the coverage threshold is still met for all its grid points. This shows how DTA uses cooperative sensing to meet the coverage threshold, enabling higher-energy IoT devices to compensate for devices with lower energy and conserve energy at devices when the threshold is already met by upstream neighbors.

The energy per epoch (see Fig. 10e) shows a recurring sawtooth pattern during day 2, between epochs 1-8 when the devices intermittently power down. This is because, within an epoch, the battery charges enough to power up the device momentarily. But, after this stored energy is immediately allocated to task execution, the battery is depleted, leading to device shutdown. As more solar energy becomes available during the day, the battery can be charged more, thus mitigating this sawtooth pattern. However, with DTA (dotted lines), the total energy per epoch remains relatively fixed and stable. Experimental results also show energy savings at the IoT devices range from 0.16% to 36% with an average of 5%.

We compare the grid point coverage per epoch with DBS with and without DTA (see Figs. 11a- 11c for the image task results). Similar results were seen with all tasks. Red and green indicate under-covered and over-covered grid points respectively. Blue shows grid points that are at the target coverage threshold. Using DTA (task adaptation) significantly reduces over-coverage (green) and under-coverage (red), providing the highest number of grid points at the coverage threshold (blue). In terms of percentages, for the image task, DBS without DTA shows 53.5% of grid points are under-covered while 46.6% are over-covered. DBS+DTA reduces under-coverage and over-coverage to 20.9%, and 10% respectively, and increases the percentage of grid points at the coverage threshold to 69%.

C. Simulation Experiments - Large Network Results

We used a non-uniform deployment pattern in a 100mx100m grid, 100 sticks and $C_T = 9$. Due to the network size and centralized scheduler runtime we only compared DBS with the Random and Alternate schedulers and enabled the DTA for all. We evaluated the performance during cloudy and sunny weather. Energy saved per device on sunny days is higher than on cloudy days due to the higher initial task values. On average, adaptation yielded per-device energy savings of 3.34% on cloudy days vs. 38.53% on sunny days. The maximum energy savings observed for an individual device was 44.92% on a sunny day. We also evaluated DBS, Random, and Alternate schedulers with different C_T values. DBS always performs best, then Alternate, then Random Schedulers.

VII. Conclusion

In this paper, we presented a cooperative sensing solution that utilizes distributed coordination between multi-sensor IoT devices. Spatial redundancy is minimized (while meeting a minimum desired coverage level) through a distributed token passing algorithm (DTA). DTA utilizes a DHT and a two step (forward and reverse) token traversal process to balance energy consumption among all the devices traversed by the token. The reverse token traversal process also optimizes the exact sensing instants using our DBS algorithm, which runs in time that is quadratic in token size and minimizes temporal overlaps between sensors with overlapping coverage. For both DTA and DBS, the token size trades off computation and communication complexity with the efficiency of sensing operations. Our on-device experiments demonstrate that our algorithm coupled with a small token size requires minimal execution time and communication energy. DTA simulation results show average energy savings of 5% per IoT device in small networks under various weather conditions. Some devices with overlapping coverage areas achieve energy savings above 30%. Further, DBS consistently achieved performance close to the optimal.

References

[1] J. Adkins, B. Ghena, N. Jackson, P. Pannuto, S. Rohrer, B. Campbell, and P. Dutta, "The signpost platform for city-scale sensing," in *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2018, pp. 188–199.

[2] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman, "Farmbeats: An iot platform for data-driven agriculture," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, Mar. 2017, pp. 515–529.

[3] E. Liri, K. K. Ramakrishnan, K. Kar, G. Lyon, and P. Sharma, "Invited paper: An efficient energy management solution for renewable energy based iot devices," in *Proceedings of the 24th International Conference on Distributed Computing and Networking*, ser. ICDCN '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 20–27. [Online]. Available: https://doi.org/10.1145/3571306.3571387

[4] Z. Zhang, J. Wu, Y. Zhao, and R. Luo, "Research on distributed multi-sensor cooperative scheduling model based on partially observable markov decision process," *Sensors*, vol. 22, no. 8, 2022.

[5] S. Hemminki, K. Zhao, A. Y. Ding, M. Rannanjärvi, S. Tarkoma, and P. Nurmi, "Cosense: A collaborative sensing platform for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '13. NY, USA: ACM, 2013.

[6] S. R. Sarangi, S. Goel, and B. Singh, "Energy efficient scheduling in iot networks," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. NY, USA: ACM, 2018, p. 733–740.

[7] A. Caruso, S. Chessa, S. Escolar, X. Del Toro, and J. C. López, "A dynamic programming algorithm for high-level task scheduling in energy harvesting iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2234–2248, 2018.

[8] C. Moser, J.-J. Chen, and L. Thiele, "Dynamic power management in environmentally powered systems," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2010, pp. 81–88.

[9] M. De Benedetti, F. Messina, G. Pappalardo, and C. Santoro, "Jarvsis: a distributed scheduler for iot applications," *Cluster Computing*, vol. 20, no. 2, pp. 1775–1790, 2017.

[10] C. Delgado and J. Famaey, "Optimal energy-aware task scheduling for batteryless iot devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1374–1387, 2021.

[11] J. Yang, X. Wu, and J. Wu, "Optimal scheduling of collaborative sensing in energy harvesting sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 512–523, 2015.

[12] O. M. Gul and E. Uysal-Biyikoglu, "A randomized scheduling algorithm for energy harvesting wireless sensor networks achieving nearly 100% throughput," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 2456–2461.

[13] E. Liri, K. Ramakrishnan, and K. Kar, "Energy-efficient distributed task scheduling for multi-sensor iot networks," *IEEE Network*, vol. 37, no. 2, pp. 318–324, 2023.

[14] S. He, K. Shi, C. Liu, B. Guo, J. Chen, and Z. Shi, "Collaborative sensing in internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2022.

[15] E. Liri, P. K. Singh, A. B. Rabiah, K. Kar, K. Makhijani, and K. K. Ramakrishnan, "Robustness of iot application protocols to network impairments," in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2018, pp. 97–103.

[16] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[17] ANSI, *Local Area Networks: Token Ring Access Method and Physical Layer Specifications–802.5*. USA: John Wiley & Sons, Inc., 1985.