Enabling Environmentally-Powered Indoor Sensor Networks With Dynamic Routing and Operation

Jia Guo, Teng Xu, Theano Stavrinos and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
Email: {jia, xuteng, theano, miodrag}@cs.ucla.edu

***Abstract*—Self-sustainable systems exploit the power sources in the environment (such as solar power) to maintain it's operations. Wireless sensor networks, one of the most popular applications of self-sustainable systems, may harvest environmental energy to sense and communicate. A key issue of self-sustainable sensor networks is that the power source varies constantly. Thus, it is important for the system to intelligently adjust the workload on each sensor to increase the network lifetime. In our work, we propose a dynamic routing and operation strategy for self-sustainable sensor networks. The system configuration is adjusted dynamically according to the amount of energy harvested at each sensor node. Our strategy aims to maximize the sampling and communication frequencies of each sensor while not compromising system sustainability. Experimental results indicate that our strategy enables the best system performance and a higher probability of self-sustainability compared to other baseline strategies.**

I. Introduction

Sensor networks are an emerging technology for collecting and transmitting information about an environment. One notable application of sensor networks is to measure the various environmental parameters in buildings, such as lighting, temperature, and humidity, in order to enable smart control of the heating, ventilating, and air conditioning (HVAC) systems. A sensor network usually consists of dozens or even hundreds of small, wireless connected sensing devices which until recently used batteries as their energy source. However, the reliance on batteries imposes limitations on where and how these sensor networks can be used. The number of nodes in the network and the placement of these nodes, among other things, are limited by the feasibility of replacing exhausted batteries over the working lifetime of the network.

As such, energy-harvesting nodes have emerged as a way to maintain a sensor network with little or no human intervention. Depending on the application, nodes can harvest solar, wind, piezoelectric, or other external energy, and either store or immediately use the harvested energy. With battery-powered nodes, the challenge is how to minimize energy use to delay the next battery replacement. With energy-harvesting nodes, the challenge becomes how to adapt network function in the face of uncertain or changing energy availability such that the network could sustain itself indefinitely.

In this paper, we consider a stationary (non-mobile) indoor sensor network that harvests solar power from its environment using small solar panels. Nodes use the harvested energy to sense their environment and wirelessly transmit the collected data, in a multi-hop fashion, to a sink node (e.g. desktop computer). We design a routing strategy for a sensor network which not only is energy-neutral, but which maintains sensing quality even in periods where external energy is not readily available for harvesting (e.g., overnight).

We begin discussion of our self-sustainable sensor communication strategy by identifying our major goals.

- "Sustainability": the sustainability of the sensor network is guaranteed. In other words, the energy consumed by each sensor node should always be less than the energy harvested.
- "Maximized performance": the performance of the sensor network is maximized under the premise of network sustainability. In our work, we use the sensor sampling frequency to measure system performance.

Our key strategy for achieving both goals is to adopt a dynamic network routing and operation strategy. Two key observations motivate our strategy design. The first is that at different times of day, different nodes harvest the most/least amount of energy. For example, during the daytime, the sensors near windows harvest the most energy while during the night, the sensors that are close to indoor lighting harvest the most energy. The second is that with different routes, the communication cost of the same sensor node varies drastically. In the multi-hop network, a node could have multiple alternative next hops. Thus when routes vary, different amounts of data are routed through the nodes, causing significant variations in energy consumption. Combining these two observations, the core idea of our routing strategy is to adjust the routes in such a way that the sensor nodes that harvest more energy take on more routing tasks, and the sensor nodes that harvest little energy take on fewer routing tasks. Therefore, all the sensor nodes in the network are more "sustainable" while still maintaining high system performance.

Compared to previous work which usually emphasizes a static routing algorithm, our major contribution is that we have split a single day into multiple segments and adaptively select a routing strategy for each segment. The routes are adjusted in such a way that the nodes which harvest more energy in the current segment take on more routing tasks. In this way, we effectively avoid the situation where nodes that harvest little energy become the energy bottleneck of the system. In addition, such a strategy ensures that the routes adapt to

978-1-5090-0733-2/16/$31.00 ©2016 IEEE

available energy and promote energy neutrality.

Another major contribution is our method of deriving the parameters for various system operations. For one, we decouple the derivation of routing and sampling frequency and apply optimization techniques separately, conquering an otherwise complex problem. In addition, we are able to obtain a unified solution despite the uncertainty of the energy harvesting among different days. Using our system configuration, the network neither spends too much energy, leaving itself unsustainable during the cloudy days with poor energy harvest; nor is it too conservative, wasting the energy harvested during days with more light availability.

II. Related Work

A. Energy Harvesting

Various energy harvesting technologies have been proposed in the past. Notable examples include solar cell, vibrational, biochemical, and motion based [1] [2] [3]. More recently, radio frequency energy harvesting has emerged as a promising technique to supply energy to wireless networks with high energy efficiency [4].

Among the various harvesting modalities, solar energy harvesting normally employs a highest power density [5]. It is a promising renewable resource considering its high output efficiency and ability to be utilized in a variety of locations. Photovoltaic production has been doubling every 2 years, increasing by an average of 48% each year since 2002, making it the world's fastest-growing energy technology [6] [7]. Many manufacturers produce low power photovoltaic devices to converting solar energy into direct current electricity, including EnOcean [8], G24i [9], IXYS [10] and Panasonic [1]. In our work, we examine solar cell based harvest technology to power the sensor network.

B. Self-sustainable Sensor Networks

Self-sustainable sensor networks rely upon energy harvesting technologies for its operations, resulting longer lifetime than their battery-operated counterparts. Varies aspects on sensor network energy harvesting have previously been studied. Paradiso et al. discussed the performance of energy scavenging for wireless electronics with different harvesting technologies [11]. Srivastava et al. proposed the power management design consideration for the energy harvesting devices. Compared to the battery-based systems, instead of minimizing the total energy consumption, energy harvesting devices operate in such a way that the energy used is always less than the energy harvested while maximizing the system performance [12]. An energy aware routing protocol on the energy scavenging ad hoc sensor networks are discussed by Shah et al [13]. Royer et al. have reviewed routing protocols for resource-limited ad hoc wireless sensor networks [14]. More recently, Xu et al. proposed energy saving schemes for medical sensor networks using sensor selection [15][16]. Eventually, a few implementations of solar energy harvesting sensor nodes are proposed including Prometheus[17], HydroWatch[18], Heliomote[19], and Ambimax[20].

While these self-sustainable sensor networks focus on designing and implementing only a single routing algorithm on the network to maximize system performance, our design emphases on dynamically changing and adopting different routing algorithms according to the amount of energy harvested in different periods of a day. We compare and present the performance of our dynamic routing scheme compared to the static routing scheme on our tested dataset in Section VIII-A.

C. Dynamic Routing for Sensor Networks

Many previous works on dynamic routing algorithms target sensor networks that are spatially dynamic (i.e. have moving nodes) rather than temporally dynamic (i.e. have nodes with changing level of harvested). Examples of algorithms in this category include Dynamic Source Routing [21], Dynamic Proxy Tree-Based Data Dissemination [22], etc.

Only a few researchers have studied the adaptive routing algorithms for environmentally powered sensor networks. Zhang *et al.* proposed a network utility function based optimization method that jointly optimizes data gathering and data transmission [23] in rechargeable sensor networks. Comparing to their approach, we used a different method where the system parameters are obtained through offline training and applied online to achieve the objective.

III. Overview

Our method consists of three steps as shown in Figure 1. The first step is to partition a day into multiple segments assuming that the optimal routing algorithm will be recomputed for each segment. A new segment should be created when the relative energy harvested by each sensor node changes dramatically. Our second step is to find an optimal system configuration which contains two parts, route selection, and sampling frequency selection. For route selection, our goal is to create a customized routing algorithm for each segment based on the amount of energy harvested by each sensor node. It is motivated by the maxflow algorithm and is designed in such a way that the sensor nodes that harvest more energy take more routing tasks. For the sampling frequency selection, the objective is to select an optimal sampling frequency for each segment such that the sensor network achieves good overall performance (high sampling frequency) while maintaining a high probability of sustainability.

We execute our method on the training data, and obtain a system configuration (the specific routes and sampling frequency corresponding to each segment). We then apply the system configuration to testing data to validate our method.

IV. System Model

A. Network Model

We consider a multi-hop network model with a single sink node. We assume that the sink node's energy and bandwidth are unlimited. Each sensor node acts both as a sensor and a router for other sensor nodes when transmitting information. In our model, each sensor node can transmit data up to

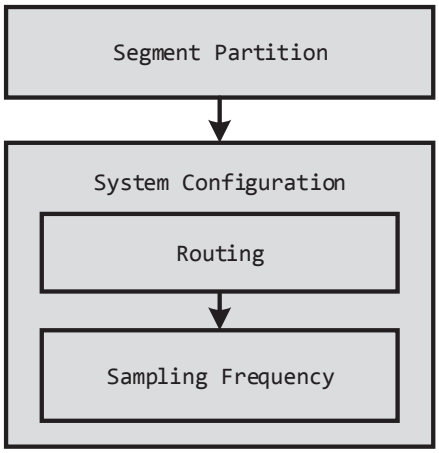

Fig. 1: System workflow.

a threshold distance, since the connectivity between sensor nodes drops dramatically past the threshold. Therefore, each sensor node must choose a path through the routers (sensors) to transmit its data to the sink node.

Figure 2 shows an example network model with ten sensors. It traces the paths from the four nodes at the bottom of the figure to the sink node. Note that each sensor not only needs to transmit its own data to the sink node but also possibly may be used as a router by another sensor node. Specifically, the sensors nodes that are close to the sink node will normally take more responsibility to route than the nodes that are far from the sink node.

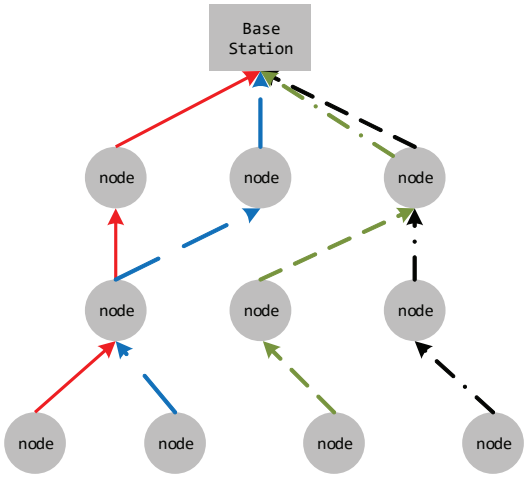

Fig. 2: An example network model with 10 sensors. Arrows represent paths from nodes to the sink node. Intermediate nodes are responsible for their own data as well as that of other nodes farther away.

B. Energy Model

Our system energy model can be classified into four sub-categories.

- *Energy harvesting model*. Each sensor attaches to a solar cell to harvest energy. The harvested power P_h is proportional to the intensity of light l_h received by the sensor's solar cell, $P_h = \alpha * l_h$.
- *Energy leakage model*. The energy storage of each sensor node leaks at a constant rate. Assume the leakage power is P_l.
- *Sensing energy model*. Each sensor powers up and obtains readings of environmental variables (temperature, light, and humidity) for time period t_s with active power P_s.
- *Communication energy model*. when two sensors communicate, the transmitter and the receiver wake up and talk for time period t_c. The power consumed by transmission is P_{ct} and the power consumed by receiving is P_{cr}.

We assume that every time a sensor takes a reading, it must transmit the collected data to the sink node. We also assume that all the sensors in the network have the same sensing and communication frequency F since all the locations in the space should be monitored equally. Then the performance of the sensor network can be measured with F, as a higher frequency indicates the environment is being monitored more accurately.

However, correspondingly, a higher frequency incurs a larger energy cost to a sensor. In order to build a sustainable network, for each sensor node i, the following equation must hold. In Equation 1, we consider time period t_1 to t_2. The left side of the equation indicates the amount of energy harvested ($\int_{t_1}^{t_2} P_h$) minus the amount of leakage energy ($\int_{t_1}^{t_2} P_l$), thus it represents the upper bound of the amount of energy for sensor i to spend. The right side of the equation has two terms. The first term represents the energy spent for sensing ($\int_{t_1}^{t_2} FP_s t_s$) and the energy spent for communication ($\int_{t_1}^{t_2} FP_{ct} t_c$) for sensor i at frequency F. The second term represents the energy spent by sensor i to be used as the router by other sensors, and we assume it is used by β times. Since as a router, the sensor needs to be used first as a receiver, then as a transmitter, the total energy spent as a router is $\beta \int_{t_1}^{t_2} F(P_{ct} + P_{cr})t_c$.

$$\int_{t_1}^{t_2} P_h - P_l > \int_{t_1}^{t_2} F(P_s t_s + P_{ct} t_c) + \beta \int_{t_1}^{t_2} F(P_{ct} + P_{cr}) t_c \tag{1}$$

C. Dataset

The Intel Berkeley Lab's public dataset provides light availability in a real-life environment[24]. The data is obtained by a 54-node wireless sensor network that collected light, humidity, and voltage measurements roughly every 30 seconds over a period of three weeks. The sensors, called motes, were battery-powered and statically arranged in the indoor lab space of the Intel Berkeley Lab at the University of California, Berkeley. There are in total over two million timestamped temperature, light, and humidity readings in the dataset.

The sensor placement is showed in Figure 3. In order to measure the whole lab environment precisely, sensors are evenly distributed in all the locations in the Research lab. Some of them are in the corner, such as sensor 42, sensor 16 and sensor 24. Others are in the office, such as sensor

54. Together these sensors provide a wide variety of indoor lighting conditions.

Before applying our algorithms, we need to preprocess the data due to missing epochs and asynchronous measurements. Our approach is to split the days into time slots. Each time slot is two minutes. If sensor i has more than 1 measurement during that time slot, we take the average of those measurements. If sensor i has no measurement on time slot t_1, we go backward and find the first time slot t_2 ($t_2 < t_1$) that has a measurement. Otherwise, we go forward and find the first time slot t_3 ($t_3 > t_1$) that has a measurement, then calculate the weighted average for t_1. The equation is showed in equation 2. T_{i,t_1} represent the temperature for sensor i at time slot t1.

$$T_{i,t_1} = [T_{i,t_2} * (t_3 - t_1) + T_{i,t_3} * (t_1 - t_2)]/(t_3 - t_2) \quad (2)$$

V. Segment Partition

Our adaptive routing and sensing strategy relies crucially on ensuring that all nodes have sufficient power to fulfill sensing and routing obligations. Nodes receive varying amounts of light during the day. Furthermore, because nodes are distributed throughout an indoor space, the amount of light that one node receives in relation to another is subject to change throughout the day.

For example, a node located near an east-facing window will receive strong light in the morning and relatively less in the afternoon, while a node located far from any windows will receive a steady amount of light throughout the day. We might want to use a different routing strategy in the afternoon to account for the now much lower energy availability of the window-adjacent node. Thus, every 24-hour period is segmented into blocks to account for the changing relationship of light availability among nodes. Each day is split into n segments; a low number of n is chosen to avoid the cost of switching the routing strategy too frequently.

By considering the light measurement of each node at an instant of time as a point in n-dimensional space, where n is the number of nodes, we can consider the centered Pearson correlation coefficient between two timepoints as a measure of similarity between the timepoints. The correlation coefficient in this case represents the cosine of the angle between rays from the origin to each of the two points in the n-dimensional space. Because our aim is to change the routing strategy only when the relationships between nodes' access to energy change, we set segment boundaries when the average similarity among points in a segment falls below a particular threshold.

Overall light availability patterns will be similar from one day to the next; our partitioning algorithm therefore considers light vectors collected at the same time of day across different days as belonging to the same timepoint. A start time is set corresponding to the end of the previous segment (the choice of initial start time is discussed below); the average similarity between every pair of timepoints from the start time to an incremented end time is then calculated, until the average similarity falls below a threshold. The end of the segment is then set to the last timepoint before the subthreshold dissimilarity occurs. The algorithmic flow to pick the start and the end point of each segment is shown in Algorithm 1.

Algorithm 1 Segmentation Algorithm

Input: *luxdata*, hourly n-dimensional vectors of light measurements from n nodes over d days; *thresh*, similarity threshold; t_0, the start point of the 24-hour period

Output: *segments*, (*start*, *end*) pairs corresponding to the start and end of the segments

```
hours ← ordered sequence of hours in the 24 hours following t_0; segments ← [ ]; lastbreak ← 0
for each hour h in hours since lastbreak do
    similarities ← []
    for each pair of hours t_1, t_2 in lastbreak to h do
        Add Pearson correlation of each vector in luxdata at t_1 with each vector in luxdata at t_2 to similarities
        if min(similarities) < threshold then
            Add (lastbreak, h − 1) to segments
            lastbreak ← h − 1
        end if
    end for
end for
Add (lastbreak, t_0) to segments
return segments
```

Because segment boundaries depend in part on other timepoints in the segment, the initial segment boundary must be carefully selected. Since there are relatively few timepoints, we select the initial segment boundary by partitioning based on all possible starting timepoints. Out of the 24 possible partitions (corresponding to a partition starting at each of the 24 hours of the day), we determine which partition gives the desired number of segments at the lowest threshold. If there are multiple such partitions, we choose the partition that starts at a stable timepoint, i.e., a timepoint at which most or all of the partitions place a segment boundary.

VI. Routing Algorithm

Based on the above partition, the next step of our optimization is to select a routing algorithm for each segment. Our algorithm design is motivated by the maxflow problem with edge constraints. We first create a directed graph $G = (V, E)$ where each node $v \in V$ represents a sensor node or a sink node and each edge $e \in E$ represents the wireless link between sensor nodes or between a sensor node and a sink node. Each node in the graph has an attribute *capacity* which describes the amount of energy available for that node. Each edge (from node i to node j) also has the *capacity* attribute which represents the amount of energy spent to transmit data from node i to node j.

To be consistent with the traditional maxflow algorithm model which only has capacity constraints for edges, we

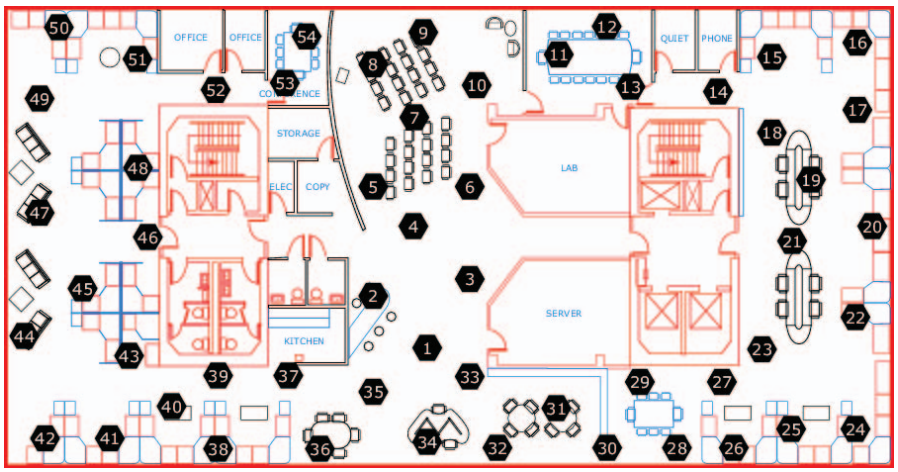

Fig. 3: The 54 sensors placement of Intel Berkeley Lab [24].

convert our graph to $G' = (V', E')$, where each node v in the original graph breaks down to two nodes v_1, v_2, as well as an edge from v_1 to v_2 which carries the original capacity of node v in graph G. After this, the node capacities are converted to edge capacities. The maxflow algorithm also requires a source node and a sink node. We virtually add a source node which has an edge connecting to all the non-sink nodes, where each edge has capacity ∞. All the data flow eventually goes to the sink node.

Figure 4 shows a four-node sensor network with maxflow graph G and the graph G' that it is converted to. C_i is the capacity for the ith sensor node in the graph, and the capacities of all solid edges are ∞. Note that the edges between the sensor nodes are mutual in the sense that node A can send data to node B while node B can also send data back to node A as long as the two nodes are within the threshold distance of one another.

In our model, we assume that all the sensors share the same sensing and communication frequency F as shown in Equation 1. The goal of our modified maxflow algorithm is to find a maximized F and the corresponding routing algorithm for each sensor node. Firstly, as different nodes harvest a different amount of energy, the capacity C_i of each sensor node is different from each other. Maxflow assigns more routing traffic to the nodes that harvest more energy. Thus, it uses as much energy as possible. Secondly, we have specifically applied the maxflow algorithm with edge constraints to ensure the energy flow that goes through each sensor node is beyond a certain threshold. Note that the energy flow of node i is proportional to the frequency F.

The core idea behind maxflow algorithm with edge constraints is that each edge e not only has a capacity $c(e)$ as the upper bound of flow $f(e)$, but also a capacity $d(e)$ as the lower bound such that $d(e) \leq f(e) \leq c(e)$. We have modified and applied it to our model. Before explaining the detail of our algorithm, we define the following two notations as shown in Equation 3. E_{start} denotes the energy spent on node i to sense and to send messages while E_{router} denotes the energy that is spent on each router nodes that routes the message from node i.

$$\begin{aligned} E_{start}(F) &: F \int_{t_1}^{t_2} (P_s t_s + P_{ct} t_c) \\ E_{router}(F) &: F \int_{t_1}^{t_2} (P_{ct} + P_{cr}) t_c \end{aligned} \tag{3}$$

Our routing algorithm is shown in Algorithm 5. In order to calculate the largest possible F, our algorithm has three main steps. The first two steps are to modify the graph, and the last step is to apply the maxflow algorithm with constraints. We first update the original capacity C_i for sensor node i with $C_i - E_{start}(F) + E_{router}(F)$ with the goal of allocating the sensing cost for each sensor node first. We intentionally add the router cost E_{router} to the modified capacity because in the next step, the maxflow algorithm will deduce $E_{router}(F)$ for all the nodes in the routing path including the node i itself. The added $E_{router}(F)$ compensates for it. Then in the second step, we set the edge constraints. We only put constraints on edges that connect the source to each sensor node. Assume an edge e connects the source with node i. Then the constraint for edge e is set to $E_{router}(F) \leq f(e) \leq \infty$. This guarantees that there exists a flow of at least $E_{router}(F)$ from node i to the sink node. The sensing message from sensor node i can thus be transferred to the sink node. The third step is simply to run the maxflow algorithm with edge constraints. In our case, we have applied algorithm from [25] on our modified sensor network graph. Finally, combined with all the three steps described above, we also apply binary search to find the maximum F that can be supported by the sensor network.

VII. System Configuration

Our primary goal is to derive a strategy where the sensor nodes sample as much as possible while maintaining sustainability. That strategy can be characterized by two components: the best set of routes S_i for each segment i in a day; and the sampling frequency f_i and the corresponding communication

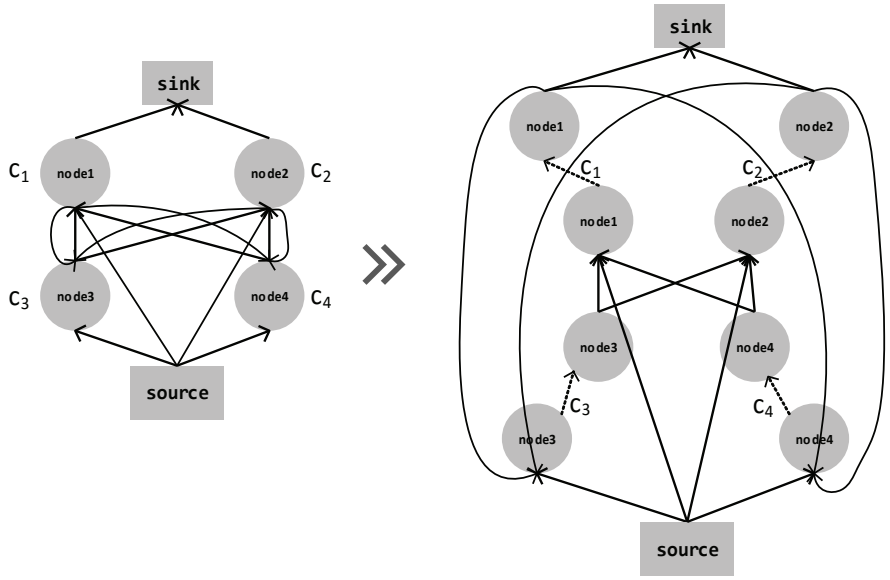

Fig. 4: An example of maxflow graph G and its converted form G'.

Fig. 5: Routing Algorithm

Input: Graph $G' = (V', E')$. Each sensor node i maps to nodes n_i, n_i' and edge $e_i(n_i \rightarrow n_i')$ in G'. The original capacity of e_i is c_i. According to Equation 1, $c_i = \int_{t_1}^{t_2} P_h^i - P_l$. The source node is s, and the sink node is t.
Output: Maximum frequency F, routing algorithm S.

1: $S = \varnothing$, $F = 0$, $F' = 0$.
2: **for** sensor node n_i in V' **do**
3: $0 \leq f(n_i \rightarrow n_i') \leq c_i - E_{start}(F)$
4: $E_{router}(F) \leq f(s \rightarrow n_i) \leq \infty$
5: $S \leftarrow$ Routing corresponding to maxflow for G' with current constraints
6: **end for**
7: **if** maxflow can be found **then**
8: $F' = binary\_search(F + \Delta F, F_{max})$
9: **else**
10: $F' = binary\_search(F_{min}, F - \Delta F)$
11: **end if**
12: **if** $|F - F'| <= \Delta F$ **then**
13: **Return** F, S.
14: **else**
15: $F = F'$, go back to step 2.
16: **end if**

frequency for each segment. Our routing algorithm produces one set of routes for each node and a corresponding maximum sampling frequency, given the amount of energy stored by each node. But we cannot plug in the expected energy harvested and use the output directly as our strategy. For one, there is variability in energy harvested among different days. For another, the set of routes that maximizes energy consumption for the current segment does not necessarily benefit the subsequent segments. The ideal strategy achieves the best results over all the segments.

To deal with the difficulties mentioned above, we start by generating different sets of routes for each segment in a day. We exhaustively run all combinations of different routes on a week's worth of data, and select the best set of routes based on the maximum sampling frequency they allow for. After that, with a fixed set of routes, we use an iterative Monte Carlo method to determine the best sampling frequency. In that way, we are able to select a strategy that yields the highest sampling frequency while being able to self-sustain.

A. Route Selection

. In the routing algorithm identification phase, we will discover a set of routes $(S_1, S_2, \ldots, S_M)$ for each segment in the day (for M segments per day). As we observe from the dataset, the relative amount of energy harvested among nodes changes over different time segments. Thus, an inconsequential node in the current time segment might become indispensable in the next, thereby requiring us to save its energy for the benefit of the following segment. Therefore, an optimal maxflow solution given the current time segment may not be optimal when we look at the global picture. To deal with the problem, we need to consider different solutions to the maxflow problem, and experimentally determine the best way to route the nodes. In practice, by randomly picking nodes and shrinking their capacity N times, we are able to produce different approximate solutions to the same maxflow problem, denoted by $\mathbb{S}_i = \{S_{i1}, S_{i2}, \ldots, S_{iN}\}$. In these different solutions, the various "alternative routes" of the maxflow problem are explored.

After running the algorithm for each segment, we will have N different sets of routes for each segment. We then exhaustively combine them into a set of solutions

$\{(S_1, S_2, \ldots, S_M) | S_i \in \mathbb{S}_i\}$ We test all the solutions with the expected energy harvest, and the one that produces the maximum overall sampling frequencies will be selected.

B. Sampling Frequency Selection

In the sampling frequency selection phase, we decide the sampling frequencies $(f_1, f_2, \ldots, f_M)$ for each segment in the day (for M segments per day). The difficulty in selecting sampling frequency lies in the variability in the energy harvested. The self-sustainable sampling frequency on one day may not be self-sustainable on another day. Yet if we simply set as standard the days with the least harvested energy, we would be too pessimistic as we ignore the energy left over from days where more energy is harvested.

We find this sweet spot by adopting an iterative Monte Carlo method. In the method, we maintain a set of "points" at each iteration t, $\mathbb{P}^t = \{p_1^t, p_2^t, \ldots, p_N^t\}$, where each point refers to one set of possible sampling frequencies, $p_j^t = (f_{1j}^t, f_{2j}^t, \ldots, f_{Mj}^t)$. We initialize f_{ij}^0 by sampling from a normal distribution $\mathcal{N}(\bar{f}_i, \sigma(f_i))$. [1] We discard points (sampling frequencies) under which the system cannot self-sustain. Then we increase the weight w_j on those points that produce more samples per day before normalizing all the weights. At each subsequent iteration t, we create a new set of points $\mathbb{P}^{t+1}$, where f_{ij}^{t+1} of each point p_j^{t+1} comes from sampling the Gaussian mixture model $\sum_j w_j \mathcal{N}(f_{ij}^t, \sigma(f_i))$. The points are then re-weighted following the same procedure as described before. In our experiment, the best point (the one with the highest overall sampling frequency) converges quickly. We will apply that best set of sampling frequencies as the settings for each of the segments.

VIII. Evaluation

Based on the sensor topology and data collected in [24], combined with the network and energy model described in Section IV, we built a simulation to validate of our algorithms. We first test the routing algorithm independently to show the effectiveness of the dynamic adjustment. Then we consider the whole system, and show the offline optimal results used as the training set. Finally, we test the configuration obtained through training in an online setting and show the sustainability as well as the performance of the system.

A. Routing Algorithms

Table I describes the maximum sampling frequency given different routing algorithms. Using the segment partition algorithm, we split a day into three segments, respectively $7am - 4pm$, $4pm - 6pm$, and $6pm - 7am$. We evaluated five routing strategies. The shortest-path method, where we always route the packet along the shortest path from the node to the sink, serves as the naive example. When we use static route methods, we do not dynamically change the routing for different segments. Instead, we compute the optimal routing algorithm on one of the segments ($1st$, $2nd$, *or* $3rd$), and apply it to all three segments. Our approach, which we call *dynamic routes*, achieves the highest overall sampling frequencies in all the segments.

[1] $\bar{f}_i$ and $\sigma(f_i)$ come from the previous step, where we select routing algorithms.

Sampling Frequency (Hz)	7am-4pm	4pm-6pm	6pm-7am
Shortest Path	0.184	0.0743	0.0104
Static Routes (1st)	0.369	0.167	0.0113
Static Routes (2nd)	0.285	0.134	0.0113
Static Routes (3rd)	0.204	0.0584	0.0207
Dynamic Routes (ours)	0.369	0.134	0.0207

TABLE I: Comparison of system sampling frequency with different routing strategies.

B. Training

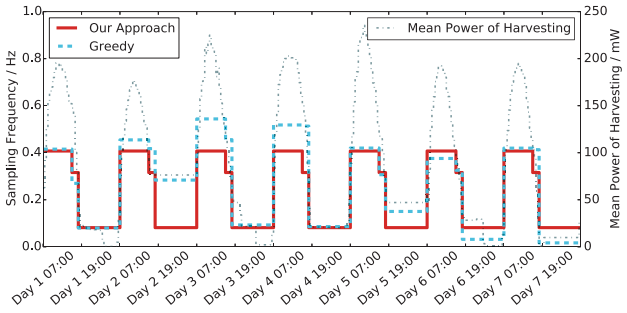

Fig. 6: The greedy and optimal sampling frequencies derived in training.

As mentioned before, we first train the system using a week's worth of data. Figure 6 shows the results. The dotted line represents the mean power of energy harvested from all the nodes. The solid line represents the corresponding sampling frequency we derive for each segment with our approach. The dashed lines represent the sampling frequency achieved by running the max flow algorithm on every segment. We refer to it as the "greedy" approach, as it always finds the locally maximum sampling frequency given the energy harvested in that segment. Apparently, the greedy approach varies from day to day. Noticeably, for the last two days, the sampling rate during the night is far below that of our approach. The reason for the variability is that some of the nodes that are bottlenecks in the communication chain did not receive as much energy as expected. Thus the overall performance is limited by those bottlenecks. In our approach, on the other hand, we execute the same strategy, derived by considering both the good and the bad days.

C. Testing

Figure 7 shows the results testing our approach on a different set of days. The dotted line shows the mean harvested power over all nodes, and the solid lines are sampling frequencies obtained from training. We also apply the same set of sampling frequencies in the greedy training scenario here. The bar below shows the sustainability of the corresponding segments, where the colored segments are able to self-sustain and the non-colored segments are not. Apparently, using the greedy sampling frequencies from training, more than half of

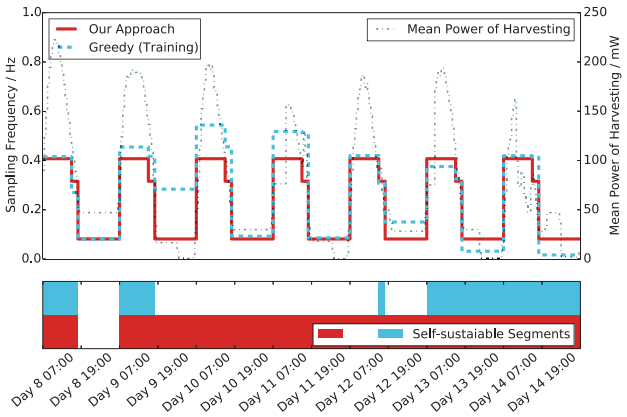

Fig. 7: Sampling frequencies and corresponding segment sustainability in testing.

the segments weren't self-sustainable as indicated by the upper part of the bar chart (9 out 21 segments are sustainable); our approach (the lower part of the bar chart) is able to sustain through almost all the segments (20 out 21 segments) [2]. The one segment that is not self-sustainable also suffered from a bottleneck node with unusual behavior. Interestingly, although the energy harvested in the last day fluctuated, both methods are still able to sustain. A possible reason for this is that most of the bottleneck nodes are far from windows while nodes near the windows are most likely to have sufficient energy supply. Thus, although the day may have been cloudy, and the nodes near windows have fluctuating energy availability, the true bottleneck nodes are not affected, therefore delivering expected service.

IX. Conclusion

We proposed a routing and operation strategy for an environmentally powered sensor network. Our approach enables the maximum workload (sensing and communicating) for the sensor network while guaranteeing the network can sustain on harvested solar energy with high probability. Our routing strategy is designed in such a way that the routing algorithm is dynamically adjusted according to the amount of harvested energy by each sensor node. We have specifically addressed our optimization flow with three steps: segment partition, routing algorithm identification, and configuration selection. Our results indicate that our dynamic routing algorithms on different segments enable the largest overall sampling frequency compared to the other routing algorithms while it can sustain 20 out of the 21 segments in our test set.

X. Acknowledgment

This work was supported in part by the NSF under Award CNS-0958369 and Award CNS-1059435.

[2]Once a node fails, we assume the whole network resets at the start of the next time segment

References

[1] "Panasonic solar cells technical handbook." http://www.solarbotics.net/library/datasheets/sunceram.pdf.
[2] S. Meninger, J. O. Mur-Miranda, R. Amirtharajah, A. P. Chandrakasan, and J. H. Lang, "Vibration-to-electric energy conversion," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 64–76, 2001.
[3] C. Melhuish, "The ecobot project." http://www.ias.uwe.ac.uk/energy_autonomy/EcoBot_web_page.html.
[4] X. Lu, P. Wang, D. Niyato, and E. Hossain, "Dynamic spectrum access in cognitive radio networks with RF energy harvesting," *Wireless Communications, IEEE*, vol. 21, no. 3, pp. 102–110, 2014.
[5] S. J. Roundy, *Energy scavenging for wireless sensor nodes with a focus on vibration to electricity conversion*. PhD thesis, University of California, Berkeley, 2003.
[6] V. Devabhaktuni, M. Alam, S. S. S. R. Depuru, R. C. Green, D. Nims, and C. Near, "Solar energy: Trends and enabling technologies," *Renewable and Sustainable Energy Reviews*, vol. 19, pp. 555–564, 2013.
[7] "Solar expected to maintain its status as the world's fastest-growing energy technology." http://www.socialfunds.com/news/article.cgi/2639.html.
[8] "Enocean." https://www.enocean.com/.
[9] "G24 power." www.g24i.com.
[10] "Ixys." www.ixys.com.
[11] J. Paradiso, T. Starner, *et al.*, "Energy scavenging for mobile and wireless electronics," *Pervasive Computing, IEEE*, vol. 4, no. 1, pp. 18–27, 2005.
[12] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.
[13] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Wireless Communications and Networking Conference, 2002. WCNC2002.*, vol. 1, pp. 350–355, IEEE, 2002.
[14] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *Personal Communications, IEEE*, vol. 6, no. 2, pp. 46–55, 1999.
[15] T. Xu and M. Potkonjak, "Energy saving using scenario based sensor selection on medical shoes," in *Healthcare Informatics (ICHI), 2015 International Conference on*, pp. 398–403, IEEE, 2015.
[16] R. Yan, V. C. Shah, T. Xu, and M. Potkonjak, "Security defenses for vulnerable medical sensor network," in *Healthcare Informatics (ICHI), 2014 IEEE International Conference on*, pp. 300–309, IEEE, 2014.
[17] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 463–468, IEEE, 2005.
[18] J. Taneja, J. Jeong, and D. Culler, "Design, modeling, and capacity planning for micro-solar power sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 407–418, IEEE Computer Society, 2008.
[19] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *Proceedings of the 4th international symposium on Information processing in sensor networks*, p. 64, IEEE Press, 2005.
[20] C. Park and P. H. Chou, "Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, vol. 1, pp. 168–177, IEEE, 2006.
[21] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*, pp. 153–181, Springer, 1996.
[22] W. Zhang, G. Cao, and T. L. Porta, "Dynamic proxy tree-based data dissemination schemes for wireless sensor networks," *Wireless Networks*, vol. 13, no. 5, pp. 583–595, 2007.
[23] Y. Zhang, S. He, and J. Chen, "Data gathering optimization by dynamic sensing and routing in rechargeable sensor networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1632–1646, 2016.
[24] "Intel lab data." http://db.csail.mit.edu/labdata/labdata.html.
[25] R. K. Ahuja, *Network flows*. PhD thesis, Technische Universitat Darmstadt, 1993.