

A Framework for Energy-Aware Scheduling

Semester Thesis

David Geiter

dgeiter@ethz.ch

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

Supervisors:

Stefan Drašković
Dr. Rehan Ahmed

Prof. Dr. Lothar Thiele

May 21, 2018

Abstract

In recent years, the popularity of all kinds of embedded systems powered by an energy harvester has drastically increased. Such devices are deployed in a large number of applications, such as data gathering, surveillance, photography, entertainment and many more. The increasing usage demand of such devices raises interests in investigations and improvements on the efficiency of these systems. The goal of this work is to build a platform that is capable of reasonable modeling of such systems and comparison of their performance under various conditions. We present the identification and abstraction of the modeled components and their functionality on the platform. In order to obtain a meaningful comparison between different systems, the platform provides various metrics, including the probability of depleting the battery.

Apart from building the platform, several experiments are performed in this work to test the platform's capabilities.

Contents

1	Introduction	4
2	Modeling	6
2.1	System	6
2.2	Execution	8
2.2.1	Modes	8
2.3	Environment	9
2.3.1	Stochastic harvesting and the steady state	10
2.3.2	Harvesting with traces	12
3	Framework setup	14
3.1	Initialization	15
3.2	Analytically-computed metrics	16
3.3	Simulation	18
3.4	Simulation metrics	19
4	Evaluation	22
4.1	Stochastic model and trace: comparison	22
4.1.1	Experiment setup	22
4.1.2	Experiment results	23
4.2	Traces: real example	25
4.2.1	Experiment setup	25
4.2.2	Experiment results	26
4.3	Varying threshold	27
4.3.1	Experiment setup	27
4.3.2	Experiment results	28
5	Conclusion	30
6	Future Work	31
6.1	Hysteresis	31
6.2	Further automation	31
A	Figures	33
A.1	Stochastic model and traces: comparison	33
A.1.1	Trace data	33
A.1.2	High vs. low traces	34

1 Introduction

Many works discuss embedded system performances in terms of energy efficiency, self-sustainability and perpetual operation. Serving as a primer example for this work, [1] provides a model for a nano-sized blimp (depicted in Figure 1) and examines on its hovering performance. Equipping the blimp with a small battery and a solar panel as energy harvester allows the possibility of a long-term operation of up to hundreds of hours.



Figure 1: *The nano blimp investigated on in [1].*

During sunny days, such a solar powered system is able to work perpetually. The battery allows the system to also operate at bad weather conditions or, depending on the size of the battery, even during night. Since the harvested energy cannot be predicted, there are several ways of modeling such a harvesting source. [1] proposes constant harvesting as well as probabilistic harvesting, where the intensity of the harvested energy is stochastically distributed. Further efficiency studies on battery equipped solar powered devices reached an analytical solution to maximize the minimum energy consumption [2], leading to a high service level being as constant as possible. Different harvesting models are proposed in [2], such as stochastic harvesting (based on historical data) or energy estimator based on an astronomical model.

Apart from different system setups and harvesting structures, the executed jobs on the system deliver a mentionable contribution to the overall performance. Depending on the scheduling scheme and the periodicity of a task, the autonomous operation time of the system may differ by a wide range. The possibilities to arrange a system's tasks to fulfill its purpose are very large, reaching from different scheduling, up to battery level dependent decisions whether to cancel some jobs.

To this date several studies have investigated on specific devices and system setups. As the number of system models, harvesting models and scheduling schemes grows larger, the possibilities of combinations vastly expand to an multitudinous amount. Having this many setup possibilities gives rise to the desire of comparing their performance. In order to make a statement about the performance and efficiency of different systems under the same conditions, a platform

is needed for a competitive comparison. In this work we provide such a platform in the form of a framework. Based on vector and matrix multiplication, the framework is able to simulate a system's battery charge level under different harvesting conditions and job execution conditions, such as the schedule. Focusing on the evolution of the battery charge during the simulation, the framework provides useful comparison metrics, such as the average time spent in desired ranges of battery charge or the probability of depleting the battery. Regarding the harvesting, the framework is compatible with different energy harvesting models, be it any kind of stochastic harvesting or real-life traces. Furthermore, the framework can deal with level-dependent battery properties, such as leakage or thermal behavior.

A crucial point for the creation of this framework is the choice and abstraction of the important parameters of a system's battery, a harvesting model and a job execution. Section 2 fully describes the scope and the modeling of the components named above. The functionality of the framework with all inputs and outputs is explained in Section 3. Section 4 provides three experiments to test the framework's capabilities, including a comparison between stochastic harvesting and a real trace. An overall bottom line on the achievements in this work is given in Section 5 and Section 6 exhibits possible extensions to the framework to further increase its functionality and accuracy.

2 Modeling

Building a framework to simulate the operation of an energy harvesting embedded device requires careful modeling. As a first step it has to be decided which components are to be abstracted from the real process and their scope in the framework. We identify these components as an energy harvester and a system with its battery and the job executions. Figure 2 depicts a schematic interconnection between the main components of the framework.

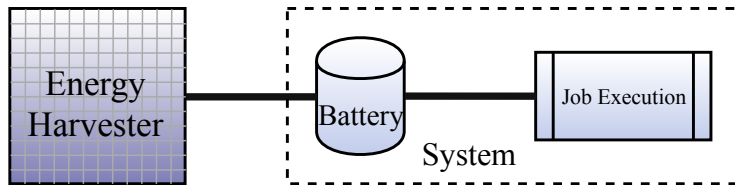


Figure 2: *Schematic of the framework's modeled components.*

The battery is charged by the energy harvester and discharged by the job executions. One of the main challenges was to identify and abstract the important parameters of the components to obtain a model comparable to the real system. We found an intuitive, simple, yet accurate way to achieve this goal. Section 2.1 describes the system model, where the battery is modeled as a Markov chain and other system properties can be specified. Section 2.2 explains the mechanics of an execution on a system and finally, the energy harvester to charge the battery is explained in Section 2.3.

2.1 System

As previously stated, one of the most important components of the system is the battery. As proposed in [1], the battery is modeled as a Markov chain where the states of the chain correspond to different discrete charge levels of the battery. The states can be seen as a one dimensional array with increasing battery level, depicted in Figure 3.

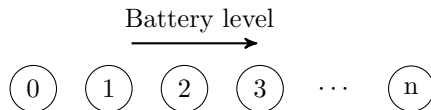


Figure 3: *The Markov chain battery with no transitions. The states of the chain represent the battery charge level.*

Charging and discharging of the battery is easily modeled with transitions between the states: charging means a transition to a higher state, discharging means a transition to a lower state. The state 0 represents the empty battery which means that the system stops operating, when entered. This state is an

important element for the framework metrics, because it indicates a system failure. In the framework, the battery takes the form of a horizontal vector \mathbf{b}^T whose values \mathbf{b}_i^T represent the probability that the battery is charged at state i . Every charging or discharging process can be described in a matrix \mathbf{T} which is the transition matrix of the Markov chain. Entry (i, j) is the probability of going from state i to state j , \mathbf{T} is therefore a square and stochastic matrix (every row sums up to 1) with dimension $n \times n$ where n is the number of battery states. Therefore, entries above the diagonal of \mathbf{T} represent charging of the battery, entries below the diagonal discharging. The battery state after such a process is calculated by

$$\mathbf{b}_{t+1}^T = \mathbf{b}_t^T \cdot \mathbf{T}. \quad (1)$$

Because of this model based on matrix multiplication, the system operates in discrete time. Figure 4 shows an example for such a charging and discharging process on a Markov chain battery. The red arrow represents a task execution where battery is consumed (see Section 2.2), the green arrows embody stochastic harvesting (described in Section 2.3).

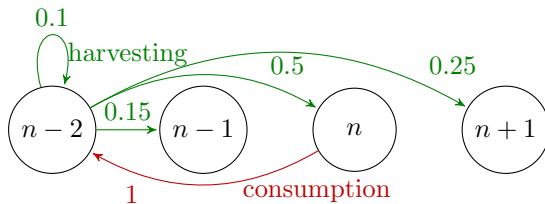


Figure 4: *The battery modeled as a markov chain. [1]*

The main advantage of using a Markov chain is that it allows us to model state-dependent behavior. For example, if we deal with battery leakage, higher states experience a higher amount of leakage.

Another property of the system is to set certain battery thresholds which splits the system in multiple sections. Each section corresponds to a mode of operation, meaning that the individual tasks may have different execution behavior (see Section 2.2.1).

Furthermore different systems may handle the same task differently, for example because of special hardware properties. The abstracted parameters of a task (computation time and power consumption on a unit processor, see Section 2.2) are therefore mapped to the system-dependent parameters using a consumption function, another useful property of the framework's system model.

It can also be specified whether the system can recover from battery depletion or if it experiences an unrecoverable failure. In the case of a persistent failure, the system model includes an absorbing error state making it impossible to charge the battery again once it has run out.

$$[0 \ 1 \ 0 \ 0] = [0 \ 0 \ 0 \ 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 5: *An execution on a full charged battery with 4 energy states in total. The executed task consumes 2 energy states.*

2.2 Execution

In the framework, an execution means the abstraction of executing a single job, an instance of a periodic task. To keep the abstraction as flexible, simple and accurate as possible, we decided to abstract the execution with two parameters: computation time and the power consumed during execution. Multiplying these attributes outputs the amount of energy consumed by the task, i.e. the discharging of the battery. Executing such a task is a deterministic process. This makes it very easy to incorporate the process in a transition matrix to model the discharge on a battery. Every entry in the matrix is zero except for one of the diagonals in the lower triangle. This diagonal is filled with ones and it is located in a distance from the main diagonal corresponding to the energy consumption. If the consumption is higher than the remaining battery charge, the battery runs out and the matrix maps to the lowest state. This truncation, proposed in [2], is represented with ones in the first column of the consumption matrix until the diagonal starts. Figure 5 shows an intuitive numerical example of equation (1) with a 4 state battery where the execution consumes 2 states from a full charged battery.

In general, if job i consumes m energy states of the battery, the resulting consumption matrix can be seen in Figure 6a. Figure 6b depicts another exemplary execution on a battery with 50 states where 7 energy states are consumed.

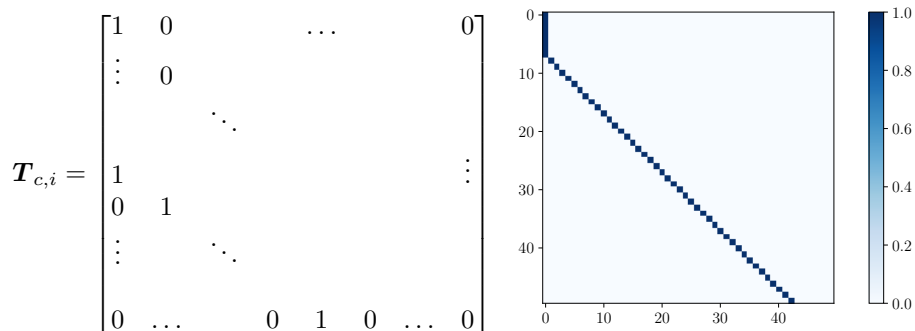
The tasks are assumed to have a certain periodicity. The smallest period where a repeating pattern over all tasks occurs is called the hyperperiod. It will be of importance when calculating the battery steady state distribution in Section 2.3.1.

2.2.1 Modes

As mentioned in Section 2.1, every task can operate in different modes depending on the current battery charge. Unifying the battery states of the same modes results in a sub-chain of operation modes. Figure 7a depicts this sub-chain, whereas Figure 7b shows a more intuitive battery schematic including the introduced operation modes.

This sub-graph provides important information for the framework metrics, such as the average time spent in each operation mode.

Here are two use-cases for such an operation mode property:



(a) General form of the consumption matrix
 (b) Exemplary execution on a battery with 50 states where 7 energy states are consumed.

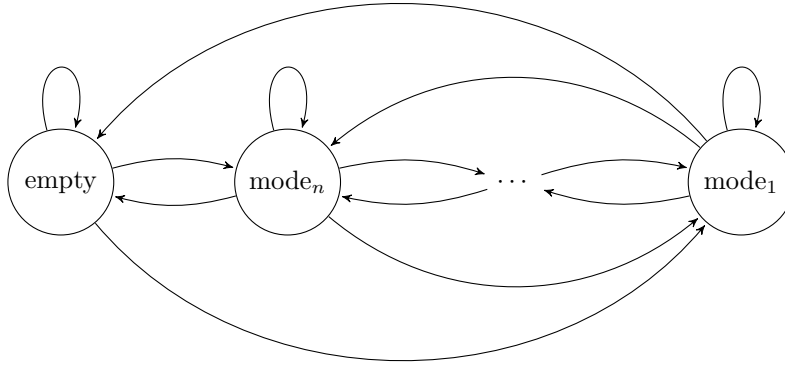
Figure 6: The consumption matrices used by the framework to model a job execution.

Use-case 1: The most obvious use-case is a degraded operation with less energy consumption when the battery charge is low. The modes are modeled as a scaling factor of the energy consumption in every battery section given by the thresholds of the system. As an example on a 50 state battery, Figure 8a shows normal execution above 50% battery charge, a degraded execution with half the energy consumption between 25% and 50%, and no execution below 25%. It can easily be seen that the submatrix in the lowest mode is the identity matrix.

Use-case 2: Another important ability for a task with modes is to execute only in very specific situations, e.g. only between two thresholds. In that case, the energy consumption would be scaled with zero for all other threshold sections of the battery as an analogy to not being executed. Figure 8b shows the case when a task executes only between 20% and 40% battery charge. An example for this could be a recovery task which does some necessary work before the battery depletes completely.

2.3 Environment

The environment encompasses everything around the system, meaning for our framework exclusively the energy harvesting. The harvesting is modeled as a vector \mathbf{v}_h with k components describing the battery charging behavior where the i -th component $\mathbf{v}_{h,i}$ of this harvesting vector represents the probability of gaining i energy states. In other words, the first entry $\mathbf{v}_{h,0}$ of the vector



(a) Sub-chain of the battery including different modes. Certain parts of the battery with neighboring nodes form a new mode state.



(b) Battery schematic including modes.

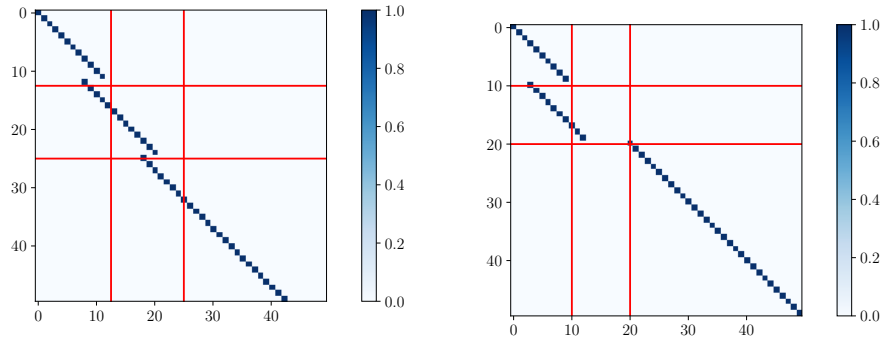
Figure 7: The battery is split up in sections of different operation modes.

contains the probability of gaining no energy, the second entry $\mathbf{v}_{h,1}$ contains the probability of being charged by one energy state, and so on. From this vector it is easy to generate a harvesting transition \mathbf{T}_h matrix for our Markov chain battery. The vector is inserted in every row where the first vector element aligns with the diagonal elements of the matrix. Like that, every state experiences the same charging behavior. If the battery would be charged to a state higher than its maximum capacity it is truncated at its maximum state (as proposed in [2]), since charging a full battery leads again to a full battery.

The framework can work with two different types of harvesting: stochastic harvesting and real traces. The following two subsections explain the functionality of these types.

2.3.1 Stochastic harvesting and the steady state

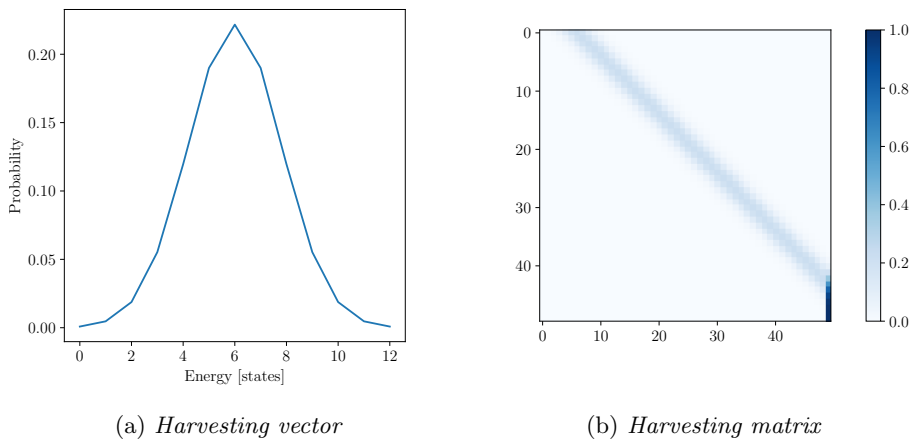
Stochastic harvesting means that the harvesting vector explained above becomes a stochastic distribution. See Figure 9 for a normal distributed harvesting vector with a length of 13, variance 0.3 and the corresponding harvesting matrix generated from the vector for a battery with 50 states.



(a) Below 50% the task executes in degraded mode, below 25% there is no execution

(b) Task execution only between 20% and 40% battery charge

Figure 8: Two use cases of degraded modes with their thresholds.



(a) Harvesting vector

(b) Harvesting matrix

Figure 9: A harvesting vector with its generated harvesting matrix for a battery with 50 states. Note the truncation on the bottom right.

A probabilistic harvesting distribution also leads to a non-trivial probabilistic battery distribution with probabilities of being in different states of battery charge. If the stochastic harvesting model is constant over the hyperperiod, the battery distribution will attain a steady state after some time, independent of the initial distribution. In other words, a steady state exists if the matrices have a certain periodicity (note that leakage is constant as well and therefore also

periodic). We are interested in this steady state for a long-term analysis, e.g. to make statements regarding a year without simulating a whole year.

For the steady state calculation we first compute the total matrix of the hyperperiod \mathbf{T}_{tot} calculated as

$$\mathbf{T}_{tot} = \prod_{\text{frames} \in \text{hyperperiod}} (\mathbf{T}_h \cdot \mathbf{T}_{\text{leakage}} \cdot \prod_{\text{jobs} \in \text{frame}} (\mathbf{T}_{c, \text{job}_i})). \quad (2)$$

Equation (2) embodies the battery evolution of one entire hyperperiod. This is done by applying the harvesting, the leakage and the consumption (i.e. multiplying the corresponding matrices) in this particular order for a worst case analysis. If the harvesting is applied first, the battery is charged to its highest possible level for the current frame. Since the leakage is an increasing function with increasing battery charge, it will then have the highest effect possible leading to a worst-case scenario. The consumption is state-independent and can therefore easily be applied last. Since all results are effected by the matrix order, choosing it was one of the main design choices during building the framework.

We then solve the steady state equation for \mathbf{b}_{ss}^T :

$$\mathbf{b}_{ss}^T \cdot \mathbf{T}_{tot} = \mathbf{b}_{ss}^T \quad (3)$$

$$\mathbf{b}_{ss}^T \cdot (\mathbf{T}_{tot} - \mathbf{I}_n) = \mathbf{0}. \quad (4)$$

These are no more than a set of linear equations. Since \mathbf{T}_{tot} is a transition matrix of a Markov chain, according to [3] one of the equations (i.e. one row of \mathbf{T}_{tot}) is a linear combination of the other equations and can be replaced by the additional necessary condition $\sum_i \mathbf{b}_{ss,i} = 1$ to exclude the trivial solution of $\mathbf{b}_{ss,i} = 0$ for all i .

The steady state analysis leads to some pre-simulation conclusions such as the average time spent in each battery state or the probability of changing between modes. Find more details about the steady state metrics in Section 3.2.

2.3.2 Harvesting with traces

We are not only interested in the probabilistic behavior of battery depletion or operating in degradation. It is also beneficial to know how the system would have performed on a certain day in the past. For this case we use recorded data traces from past days. Figure 10 shows an example for a light harvesting trace over five days in the room G78.1 of the ETZ building (Gloriastrasse 35, 8092 Zrich), a facility of the ETH Zrich.

In this case, the simulation is no longer stochastic but rather deterministic, since for every step during the simulation it is exactly known how much energy is harvested and with that how much the battery is charged. For every instant of time the harvesting vector v_h becomes a vector of length $n_h + 1$ (n_h being the energy charge intensity) filled with zeros except for the last value $v_{h, n_h - 1}$ which

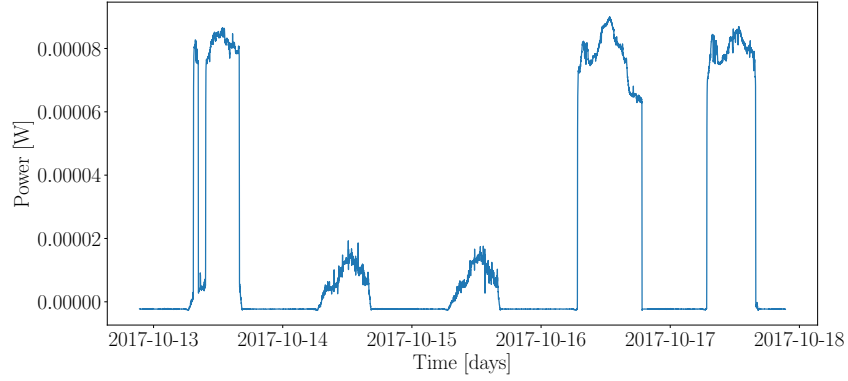


Figure 10: *A 5 day trace of light energy harvesting. Location: office at Gloriatrasse 35, 8092 Zrich.*

is one, meaning that we have a probability of 100% of charging the battery for n_h states.

After the simulation it can directly be seen at which daytime the system entered a degraded mode or stopped working due to an empty battery.

3 Framework setup

This section explains the functionality and usage of the framework. Consider the diagram in Figure 11 for a general overview of the whole framework. Explanations to every diagram component are found in the following subsections.

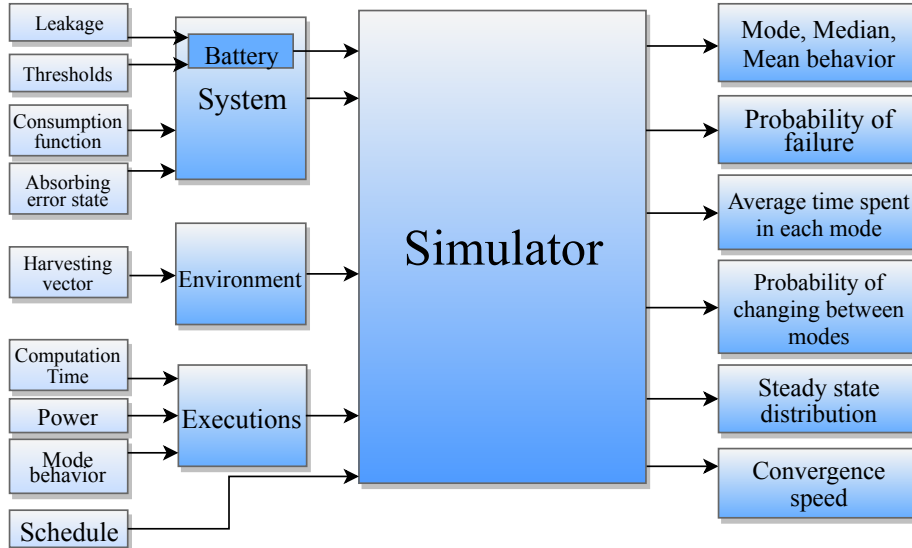


Figure 11: A schematic overview on the framework's functionality and basic features.

Since we have a discrete time simulator, we need to determine a reasonable time step for energy harvesting and leakage. We introduce this time step as a frame with the same properties as a frame in the cyclic-executive scheduling scheme, as introduced in [4]. This means that one harvesting matrix or one leakage matrix are applied per each frame with the intensity of one frame length. The tasks on the other hand possess a fixed computation time and are therefore independent of the frame length. One or several tasks can be executed per frame, as long as the frame length is not exceeded by the sum of the computation times. See Figure 12 for an example with three frames of length 4 and three tasks distributed on them. In this example, harvesting and leakage would be applied at the times 0, 4, 8 and 12.

The framework can be split up into three main parts: initialization, simulation and evaluation. The initialization contains all modeling aspects and generates specific instances of these models, the process is described in Section 3.1. Right after that, depending on the harvesting model, it is possible to gather important evaluation data through an analytic investigation, shown in Section 3.2. Then, Section 3.3 describes the actual performance of a simulation and finally, Section 3.4 provides the framework's metrics based on the simulation.

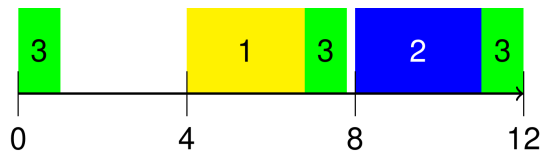


Figure 12: *An example of frames with task instances.*

3.1 Initialization

In the first section of the framework all the necessary matrices are generated with the given inputs. A crucial input for the system is the declaration of the leakage amount in each state. This is given in a text file where each leakage amount is written on a new line. This file will determine the dimension of the battery as well as the leakage behavior, making it one of the most important parts of the system instantiation. Furthermore, the individual consumption function is defined and given to the system as another input. The choice of non-increasing thresholds and decision whether to include the absorbing error state are finally needed to instantiate the system. For demonstration purposes, we generate a sample system with 100 battery states, linear slowly increasing leakage, one threshold at 25% battery charge and no absorbing error state.

The next step is to generate the tasks. This is done by instantiating each task with its computation time, power and the behavior in different modes. If no modes are given, the task execution behavior is mode invariant, which means unscaled energy consumption (remember that an operation mode scales the energy consumption). We call this the normal mode. If the number of modes is less than the number of thresholds in the system, the task won't execute in the undefined modes. With the schedule in the form of a look-up table, the taskset for the system can now be generated with its consumption matrices. The look-up table maps each task instance to its frame. Our sample system executes an arbitrary taskset with 1-2 tasks per frame (energy consumption differs by a factor of at most 4) and a hyperperiod of 6 frames. Below 25% battery charge, the energy consumption halves.

Finally the harvesting model for each frame has to be defined. This is done by initializing one or multiple harvesting vectors and giving the intervals (in number of frames) in which the corresponding harvesting model shall be constant. So in the case of throughout constant harvesting, there is only one harvesting vector and the interval is the whole simulation. When using traces, the harvesting vector changes every frame, so the intervals will only be the length of one frame. The current version of the framework supports generation of a normal distributed harvesting vector. We initialize such a normal distributed harvesting model for the sample system with a standard deviation of 0.3 where the mode (most likely value) of the distribution is 13 states.

3.2 Analytically-computed metrics

There are several parameters of interest to compile a reasonable analysis and evaluation on the performance of a system. In the case of a stochastic harvesting model, some of them can analytically be obtained before the simulation. First of all, as described in Section 2.3.1, the total matrix \mathbf{T}_{tot} over the hyperperiod can be obtained and equation (3) can be solved for \mathbf{b}_{ss}^T to obtain the battery steady state distribution. Our sample system attains the steady state depicted in Figure 13.

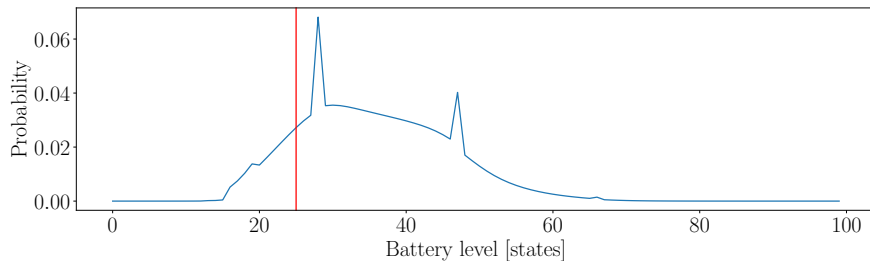


Figure 13: *An exemplary steady state distribution on a 100 state battery. The red line indicates the battery’s threshold at 25% of charge.*

The plot contains spikes in the probability distribution. Their origin lies in the leakage model where sometimes two neighboring states map to the same lower state. Steady state is reached regardless of the initial distribution, as shown in the example with full initial battery in Figure 14.

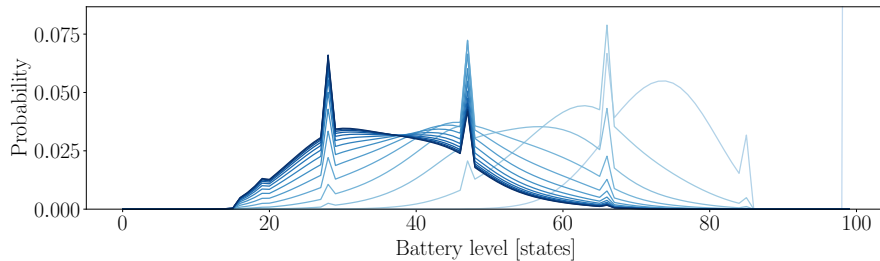


Figure 14: *Steady state convergence with a full battery as initial condition. Lighter shades represent earlier iterations.*

The color intensity of the curves is a linear rising function in time, meaning that the distributions evolve proportional to the color intensity.

The speed of convergence to a given precision can be calculated as well. Consider Figure 15 for a logarithmic plot of increasing precisions and the number

of iterations until the precisions are reached.

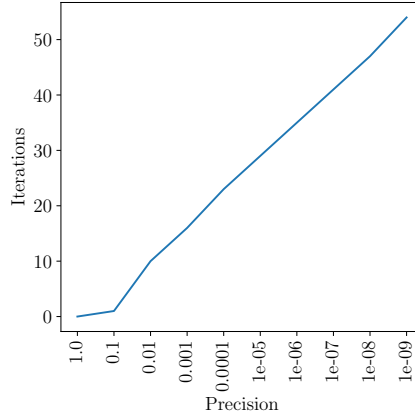


Figure 15: *The average of time spent in percentage for every state.*

The total hyperperiod matrix \mathbf{T}_{tot} also provides information about the average time spent in each state. This metric is obtained by summing each column of \mathbf{T}_{tot}^2 , like it is shown in [3]. Figure 16 shows the percentage of time spent in each state.

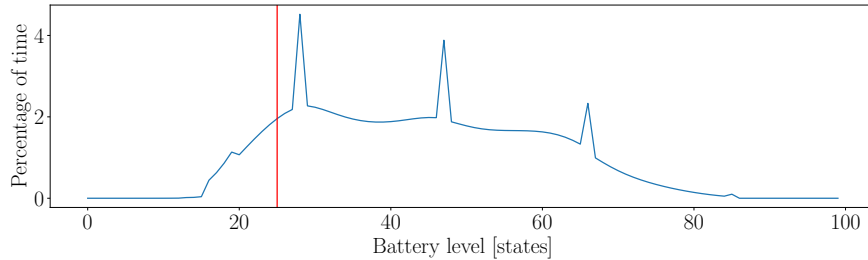


Figure 16: *The average of time spent in percentage for every state.*

We now can easily point out the average time spent in each mode, for this purpose consider Table 1. Mode 1 indicates normal operation whereas higher modes represents degradation. Mode 0 represents the empty battery. Very small values are rounded to their first appearing decimal to show that they are non-zero whereas normal scaled numbers are rounded to two decimals.

In almost 90% of the frames the system operates in normal mode, the remaining operation is in degradation. The battery is empty only for an extremely little fraction of the time. If the system would run over a year, only 28.7 milliseconds would be spent with an empty battery. Remember that constant harvesting

Modes	Time spent in percentage
0	9E-8
2	10.39
1	89.61

Table 1: *The average of time spent in percentage for every mode.*

is assumed for this analysis. Furthermore we can determine the probability of staying in a mode or changing between modes, These relations are shown in Table 2 where entry (i, j) indicates the probability of changing from the mode corresponding to the i^{th} row to the mode corresponding to the j^{th} column.

	Mode 0	Mode 2	Mode 1
Mode 0	1e-05	0.728	0.272
Mode 2	1e-06	0.512	0.488
Mode 1	5e-11	0.032	0.968

Table 2: *The probability of changing between modes.*

We see that if the system operates in normal mode (mode 1), there is a 96.8% chance of staying in normal mode. From degraded mode (mode 2) the chance is almost 50% for both, staying in degradation or changing to normal operation. Chances for battery depletion from any mode (including mode 0: being already depleted) are very small.

3.3 Simulation

The simulation section is an integral part of the framework where the battery distribution is evolved along the given number of frames. Again, as described in Section 2.3.1, we aim for a worst case analysis and apply therefore harvesting, leakage and consumption in this order before sampling the battery distribution. Depending on the initial distribution, a different evolution is obtained. Figure 17 shows a simulation of 12 frames with the steady state as an initial condition. Since the steady state is always reattained after the hyperperiod (6 frames) the evolution shows periodic behavior with deviations around the steady state. As another example, figure 18 depicts the same simulation with a different initial condition: a full battery. Since we simulate only over 2 hyperperiods, the steady state is not yet reached.

In order to see the full capabilities of the framework, we also present a simulation with a trace instead of stochastic harvesting. Figure 19a depicts the

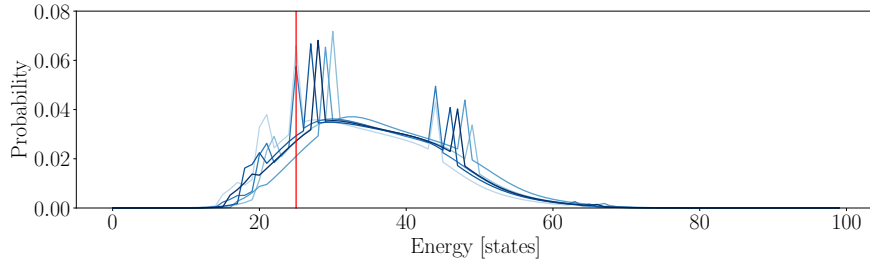


Figure 17: *Battery simulation with the steady state as initial distribution. Note the periodicity of 6 frames.*

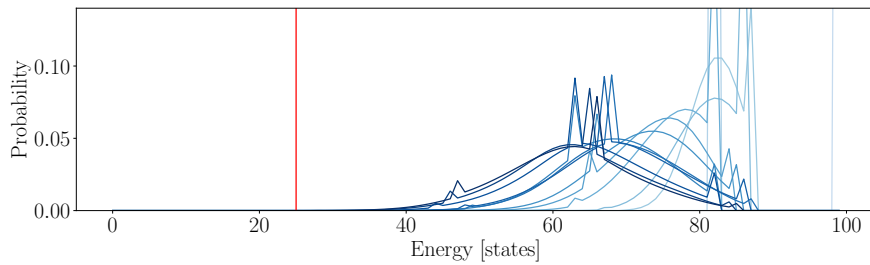


Figure 18: *Battery simulation with the full battery as initial distribution*

chosen trace and the simulation result, the evolution of the battery level can be seen in Figure 19b.

Between minutes 60 and 70, the steepness of the slope begins to cease. This is due to degradation which is entered below a battery charge of 25%.

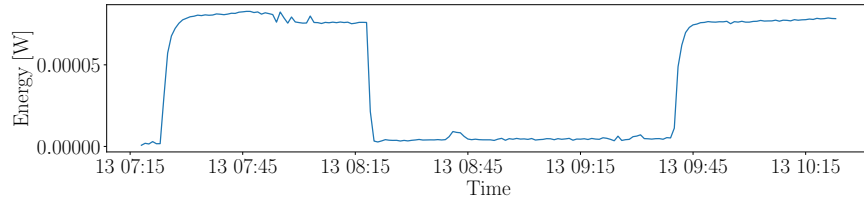
3.4 Simulation metrics

After gathering the battery distributions from the simulation, we analyze three stochastic metrics:

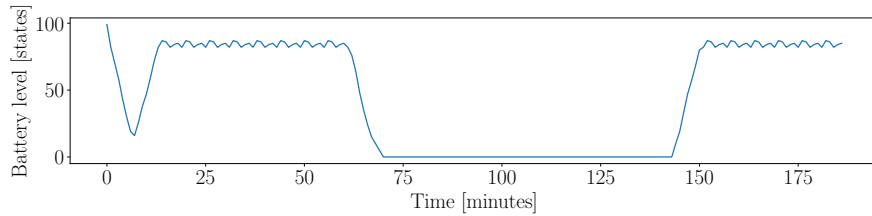
- Mode: The most likely (or most frequent) value.
- Median: The middle value. It divides the area of the distribution in half.
- Mean: The average value given by $\sum \text{value} \cdot \text{probability}_{\text{value}}$

Figure 20 shows the evolution of these metrics during the simulations with both initial conditions, the steady state (Figure 20a) and the full battery (Figure 20b).

Since the hyperperiod of the sample system has the size of 6 frames, the metrics show periodic behavior with a period of these 6 frames when starting in

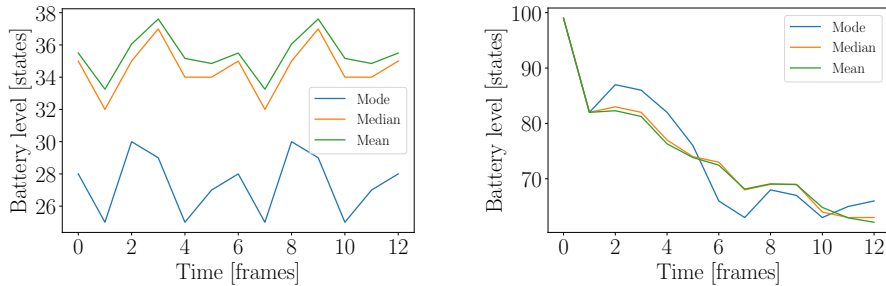


(a) *The chosen trace*



(b) *Battery evolution*

Figure 19: *Battery simulation with a trace as harvesting input.*



(a) *Steady state as initial distribution*

(b) *Full battery initial distribution*

Figure 20: *Stochastic metrics during the simulation.*

steady state. The effect of degradation is clearly visible since the most frequent value is barely above the 25% threshold whereas the median and mean are roughly 10% higher. In the simulation with initial full battery, the distribution drops sharply in the first few frames. The metrics will eventually approach the same curves as with initial steady state.

When simulating with traces, we can measure similar metrics to the ones computed analytically:

- Measured time spent in each mode
- Frequency of system failures (battery depletions)

- Frequency of transitions between modes

Comparing these measured metrics to the predicted values by a given stochastic model leads to a good opportunity of comparing a stochastic harvesting model with trace data. This will be done in Section 4.1.

4 Evaluation

Now that we have seen what the framework’s capabilities are we want to perform some experiments. Section 4.1 compares stochastic harvesting models to the traces which the models are generated from. A real use-case where a photographic sensor is modeled, proposed in [5], is provided in Section 4.2 and Section 4.3 investigates more theoretically on the impact of the threshold choice.

4.1 Stochastic model and trace: comparison

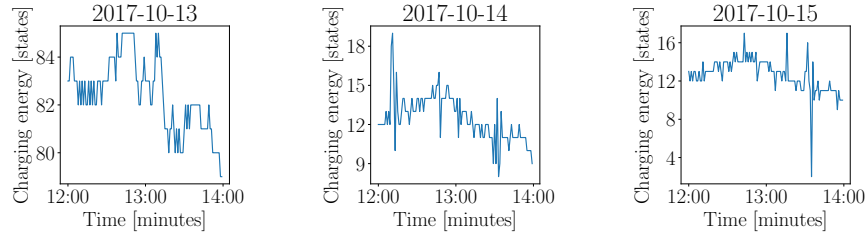
As a first experiment, the harvesting model is compared to a real trace. We therefore use a trace for the generation of a normal distributed stochastic harvesting model and run the simulation with both, the traces as well as their stochastic harvesting model. The setup is described in Section 4.1.1 and the results are presented in Section 4.1.2.

4.1.1 Experiment setup

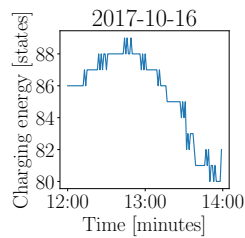
The system and taskset setup is very similar to the sample system setup in Section 3. The only difference is that the battery is equipped with 1000 states instead of 100 to increase the resolution of the harvesting. The taskset stays as before with a hyperperiod of 6 frames.

Let us get started with the trace data. We use the same data as mentioned in Section 2.3.2, the room G78.1 in the ETZ building at ETH Zrich. Out of 30 days of trace data, we use a two hour trace slice of every day which is from 12AM to 2PM, meaning we have 30 2-hour slices for the simulation and to generate the stochastic model. Figures 21a - 21d show the slices of the first 4 days, discretized in minutes. All values are scaled such that the maximum value occurring in these 30 days is set to 100 states. Non-integers are rounded down for a worst case analysis. Apparently, with a closer look at the slices, the days seem to split into two groups: high energy harvesting days, hence called high days, and low energy harvesting days, called low days. Consider Appendix A.1.1 for all 30 slices and Appendix A.1.2 for the first 10 slices compared to each other in one graph to point out the clear difference between high and low days. Motivated by this observation, we create two stochastic harvesting models, one for the high days and one for the low days. The decision for a day to be either high or low is made by checking whether its mean is above or below 50 states which is half the maximum value. After separating the days in a high and a low group we finally generate the two stochastic normal distributed models with a maximum-likelihood estimation, shown in Figure 21e.

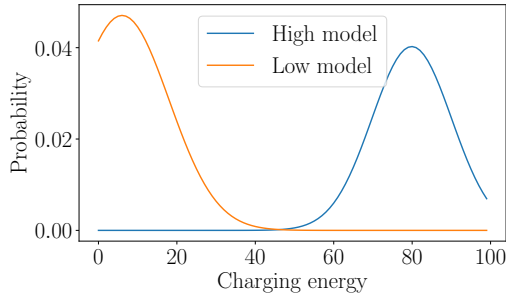
As expected, the two stochastic models clearly differ from each other leaving a gap in between them. The actual experiment takes now place as follows: We use both stochastic models to generate the analytical metrics (average time spent per mode, average probabilities of changing between modes). After that we run the simulation with all 30 traces and take the average of all high days and all low days of the metrics named above. These averages of the actual



(a) *October 13th (high day)* (b) *October 14th (low day)* (c) *October 15th (low day)*



(d) *October 16th (high day)*



(e) *Stochastic models generated from 12AM-2PM slices.*

Figure 21: *Traces from 12AM to 2PM on 4 consecutive days in October 2017. All values are normalized to the maximum value occurring in all 30 days.*

measurements are then compared to the analytical results to see whether the stochastic competes well with the real data. Every simulation starts at the mean value of the steady state distribution of its corresponding stochastic model.

4.1.2 Experiment results

As a first analysis we investigate on the average time spent in each mode. The modes of the system are the empty battery (mode 0), normal operation (mode 1) and degraded operation (mode 2) where each task consumes half the energy of the consumption in normal operation. Figure 22a compares the analytical calculated average time spent from the high stochastic model and the averaged actual measurements from all the high days whereas Figure 22b shows the same data for the low model and the low days.

The stochastic model for the high days matches the actual measurement pretty well. The model predicted that 100% of the time will be spent in normal operation (to a precision of 10^{13}) and the measured data with the traces averages at 99.86%. The remaining 0.14% were spent in degradation, below 25%

Modes	Time spent (model) [%]	Time spent (traces) [%]
0	0.00	0.00
2	1E-13	0.14
1	100.00	99.86

(a) *High days*

Modes	Time spent (model) [%]	Time spent (traces) [%]
0	21.51	97.15
2	65.81	0.69
1	12.68	2.16

(b) *Low days*

Figure 22: Comparison between the analytical average time spent generated by the stochastic model (middle column) and the average of the actual measurements (right column).

battery charge. For the low days we see a worse performance of the low model’s prediction. While the model predicted a distributed detention time over all modes (with 65.81% for degradation), the measured data provided a one-sided result with the battery being empty for 97.15% of the time.

In order to further assess the performance of the stochastic models we examine the probabilities of changing between modes. We therefore again calculate them analytically with the model data and compare them to the averaged measured probabilities. Figure 23a and 23b show the comparison of the high days, Figure 23c and 23d display the result of the low days.

Starting in normal operation (mode 1), the high model again shows good transition predictions: In most of the cases the system stays in normal operation. In a very few cases during the measurements the system entered degradation and stayed there in 50% of these cases. This differs from the predicted probabilities which state to immediately go back to normal operation, but matches the the observations about the average time spent in Figure 22a. This discrepancy can be explained by the properties of a trace: it is more likely to get similar consecutive values instead of high transients all the time, leading to a higher entry (i, i) in the probability matrices (the stochastic model is independent of the iteration before). Since the battery was never depleted during the high days, no data is provided for the first row representing the empty battery.

Let’s focus on the low days, starting with the empty battery. Here the stochastic model predicted a probability of 0.919 of staying empty and the measured data provides a probability of 0.996. Similar precision is also seen in degraded operation. We measured an almost 0.5 chance for staying there or changing to the empty mode, the model predicted probabilities of 0.605 (staying) and 0.395 (change to empty). The measured data provided very few

	Mode 0	Mode 2	Mode 1
Mode 0	0.0	0.006	0.994
Mode 2	0.0	0.0005	1.0
Mode 1	0.0	6e-15	1.0

(a) *High days - Stochastic model*

	Mode 0	Mode 2	Mode 1
Mode 0	nan	nan	nan
Mode 2	0.0	0.5	0.5
Mode 1	0.0	0.0008	0.999

(b) *High days - Trace*

	Mode 0	Mode 2	Mode 1
Mode 0	0.919	0.081	0.0
Mode 2	0.395	0.605	1e-13
Mode 1	0.0	0.409	0.591

(c) *Low days - Stochastic model*

	Mode 0	Mode 2	Mode 1
Mode 0	0.996	0.004	0.0
Mode 2	0.467	0.467	0.067
Mode 1	0.281	0.0	0.719

(d) *Low days - Trace*

Figure 23: *The probabilities of changing between modes. Entry (i, j) represents the probability of changing from the mode in row i to the mode in column j .*

transitions from degradation to normal operation as well. The model performs worse at the transition probabilities from normal operation. The prediction states a probability 0.409 of going to degradation and a 0 probability of depleting the battery, the numbers for the measured data are 0 and 0.281, respectively. Staying in normal mode matches the prediction quite well again. Note that 97.15% of the measured data in the low days stayed in the empty mode, so there was only little data available for the second and third row in the probability matrix.

4.2 Traces: real example

Now we want to address our attention to the simulation of a real use-case. [5] investigates on the performance of a solar-powered data sensor with different energy management configurations. The sensor takes environmental images and stores them on an SD card. In this experiment we simulate this sensor with trace data of 45 different days between 8AM and 8PM. The setup is described in Section 4.2.1 and the results are presented in Section 4.2.2

4.2.1 Experiment setup

[5] introduces different energy management configurations. We perform the simulations with the configuration which is used as the baseline for all comparisons. Taking an image requires 4 tasks to be executed. These tasks and their energy consumption are depicted in Table 3.

All 4 tasks are executed with a single energy burst from the battery. For our simulations the sensor takes 30 images per minute and its battery can store 834mJ of energy. We add two degraded modes which reduce the rate of taking

Task	Voltage (V_{load})	Energy (E_{burst})
Acquire One Image	3.0 V	156 μ J
Process One Image	2.0 V	527 μ J
SD Card Initialization	2.7 V	10 536 μ J
SD Card Data Transfer (per Image)	2.7 V	1 137 μ J

Table 3: *The tasks for taking an image described in [5].*

the images, such that there is more waiting time to charge the battery. The first degraded mode is entered below 40% battery charge and drops the job execution rate by 25%. Below 20% battery charge, in the lowest mode, the rate is halved compared to the initial rate. See the experiment’s schematic battery and operation modes in Figure 24.



Figure 24: *Battery schematic for the experiment. As the battery level lowers, the frequency of taking images lowers as well.*

The harvesting consists of trace data from 45 consecutive days, October 13th 2017 to November 27th 2017. Every simulation starts with the mean charge of the steady state battery distribution generated by the stochastic model of all 45 days (the same way as in the first experiment). In this way the initial battery charge attains 91%. Note that the stochastic model is generated for the single purpose of finding the initial battery charge. During the simulation we keep track of the mode in which the system operates and after running all 45 days, the proportion of each mode can be determined. The simulation is discretized in steps of 30 minutes.

4.2.2 Experiment results

Figure 25 depicts the operating modes as percental appearance in all 45 days, i.e. for each point of time the graph shows how frequent each mode was.

During the first 4 hours the system always stayed in normal operation. At 12AM (lunchtime) the system enters the first degraded mode on some days, reaching a peak fraction value of 35.56% at 2PM. After 2PM we see further degradation from the first to the second degraded mode, but the fraction of normal operations stops dropping and starts to stay constant between 57.78% and 62.22% until the end of the day. After 6PM, when a lot of people left the office, the fraction of battery depletions rises up to 31.11%.

Since we are using the same trace data set as in the first experiment, we know that there are high and low harvesting days. With that fact given, we can interpret that the system always operates in normal mode during the high days and slowly enters the degraded modes during the low days. Since we are starting every simulation with an almost full battery, we don’t see any degradation in the first 4 hours, even for the low days.

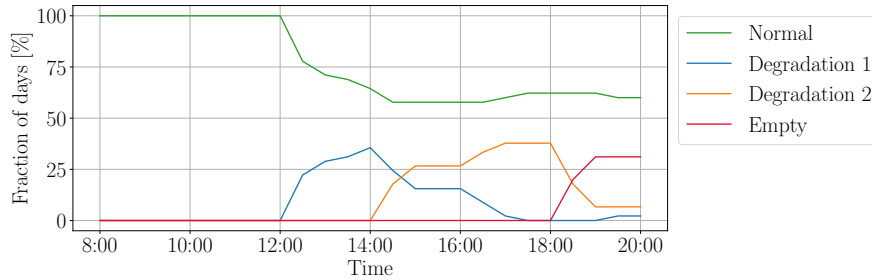


Figure 25: *The percentage of days the system operated in each mode. In degradation 1 the job execution rate drops to 75%, in degradation 2 to 50% of the initial rate.*

4.3 Varying threshold

The third experiment shall give insights on the impact on battery depletion and operation in degraded modes under varying the mode threshold. The setup is described in Section 4.3.1 and the results are presented in Section 4.3.2.

4.3.1 Experiment setup

The taskset for this experiment consists of ten tasks with linear rising energy consumptions where the lowest and the highest consumption vary by a factor of 10. There will be two thresholds splitting the battery into four parts: the normal mode, the empty mode, and two degraded modes. The higher threshold will always be two times as high as the lower one to reduce the threshold variation to one free parameter. Degradation implies no execution for some tasks, reducing the number of executed tasks in the lower modes. In normal operation mode, all of the ten tasks are executed. After passing below the first threshold of the battery level, the number of executed tasks reduces to five and below the second threshold (in the lowest mode before battery depletion) two tasks remain to be executed (see Figure 26).

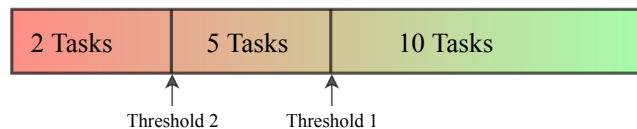


Figure 26: *The system battery with the corresponding operation modes in this experiment's setup. The thresholds will be varied. Threshold 2 is always $\frac{\text{Threshold 1}}{2}$.*

No absorbing error state is assumed, meaning that the battery can be recharged after depletion and does not lead to a persistent failure of the system. We as-

sume constant stochastic harvesting. The consumption and harvesting intensity are tuned such that with no degradation we have a battery depletion probability of 50%. As a first step the steady state battery distribution for a random schedule is calculated. We use 8 frames with 1-4 tasks per frame. Finally the actual experiment simulation is carried out: Each task is executed (always starting from steady state) and the resulting battery distribution is analyzed. We are interested in two kinds of failures:

- Battery depletion: The probability that the execution of each task leads to an empty battery.
- Degradation: The probability that certain tasks are not executed, meaning the probability of being in one of the two degraded modes.

Finally we take the maximum value for depletion and degradation over all tasks for a worst case execution scenario. This procedure is performed for threshold levels between 0% and 100% (0% and 50% for the lower threshold, respectively).

4.3.2 Experiment results

The battery states representing the degraded modes can be seen as a buffer zone between normal operation and the empty mode. Stopping certain executions in the degraded modes has a buffering effect on entering the empty mode. As this buffer zone increases in size (with rising thresholds), the battery depletion probability will therefore lower. At the same time, since more tasks stop their execution if the degraded modes expand, the degradation probability increases as well. Figure 27 depicts this relation between threshold and battery depletion probability and degradation probability, respectively.

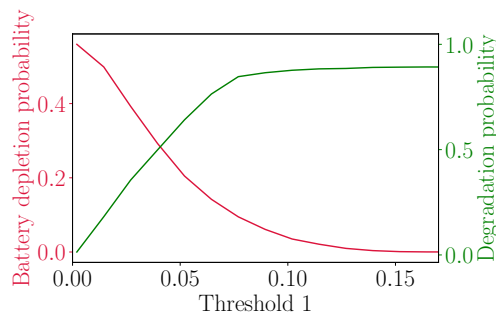


Figure 27: *The impact of varying the threshold on the battery depletion probability and the degradation probability.*

As the threshold increases the probability of battery depletion drops relatively fast and the degradation probability rises with an similarly steep slope. With thresholds at 20% and 10%, the failure probability is in the magnitude of

10^{-6} and the degradation probability is at 0.893. This means that the introduced buffer has a very strong effect, in most of the cases the system operates in a degraded mode (some tasks are dropped) which successfully prevents battery depletion.

5 Conclusion

In this work we present a framework capable of comparing the performance of different system setups, harvesting models and scheduling schemes. A Markov chain is used to model the battery, allowing us to include state-dependent properties such as leakage or thermal behavior. For the modeling of the energy harvesting, the user can choose between a stochastic harvesting model or a real trace for evaluating a system's performance on a certain day in the past. The tasks are abstracted with computation time and power dissipation, the product of these values equals the consumed energy from the battery. Due to state-dependent battery dynamics, one can define different modes for each task, lowering its computation time or shutting the task off completely on certain battery levels. The tasks can be initialized in the desired schedule.

The variety of the framework allows flexible comparisons between all components named above. For instance, different execution times (with adapted power) or individual battery dynamics can be compared. Precisely speaking, the framework provides the following metrics for an evaluation:

- Stochastic harvesting:
 - Steady state battery distribution
 - Speed of reaching steady state
 - Average time spent on each state/mode (including depleted battery)
 - Probability of changing between modes or staying in a mode
 - Mode, median and mean behavior of the battery distribution
- Harvesting with traces:
 - Measured average time spent in each mode
 - Measured frequency of changing between modes or staying in a mode

6 Future Work

During the process of careful planning and setting up the framework’s components, some features are left to be done in the future. In this section we propose possible extensions for a future reader, in order to increase the framework’s capabilities and accuracy. Section 6.1 proposes a hysteresis mode behavior to achieve a greater flexibility when working with different operation modes. Section 6.2 elucidates possibilities of reducing manual work with proposals for further automation.

6.1 Hysteresis

We presented a system model with a battery that is capable of executing task in different modes dependent on the current battery level. As a next step, one could consider the transients between these modes and differentiate whether the system enters a mode from a higher or from a lower battery level. This reasoning results in the concept of a hysteresis region for every mode transient that behaves differently if entered from lower or higher battery levels. Consider Figure 28a for a battery schematic with the hysteresis extension for one degraded mode. The battery now has more states than before, since differ between degraded operation after normal operation and degraded operation after being empty. We call those degraded sections deg_+ and deg_- , where deg_+ represents degradation entered from normal operation and deg_- when entered from a depleted battery. The underlying sub-chain resulting from the hysteresis property can be seen in Figure 28b. Some transitions are now missing, since it is not possible to change from deg_+ directly to deg_- and vice versa. Another missing transitions are the ones from the empty state to deg_+ and from normal operation to deg_- . The resulting transition matrix is shown in Figure 28c, where it is clearly stated which sub-matrices are the zero matrix of appropriate dimension.

The matrices \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} remain unchanged, since they do not map to or from a degraded mode. The degraded modes deg_- and deg_+ encompass the same number of states. To obtain the probability of being in state $i \in \text{degradation}$, one has to add the i^{th} states of both deg_- and deg_+ .

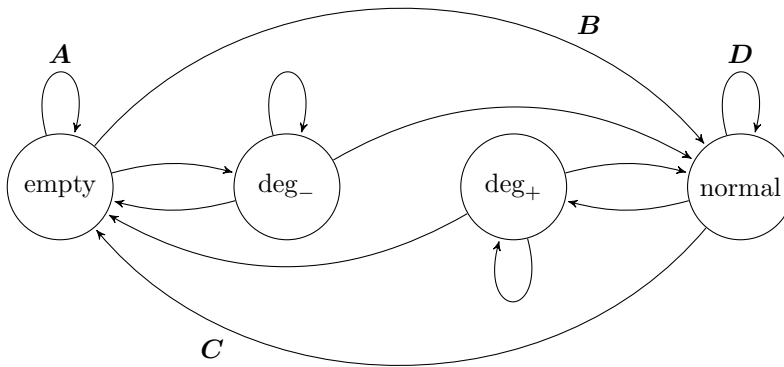
The example presented in Figure 28 contains one degraded mode. Note that this concept may even be extended to n modes with hysteresis behavior.

6.2 Further automation

At the current version, the user has to manually input the schedule in form of a look-up table that maps the task instantiations to the frames. It would be conceivable to implement functions that output a desired scheduling scheme. Another extension for further automation would be the ability to generate a larger diversity of harvesting vectors. These features would further increase the usability and flexibility of the framework.



(a) The impact of varying the threshold on the battery depletion probability and the degradation probability.



(b) The possible transitions when the degraded mode has hysteresis characterization.

	empty	deg_	deg+	normal
empty	A	*	0	B
deg_	*	*	0	*
deg+	*	0	*	*
normal	C	0	*	D

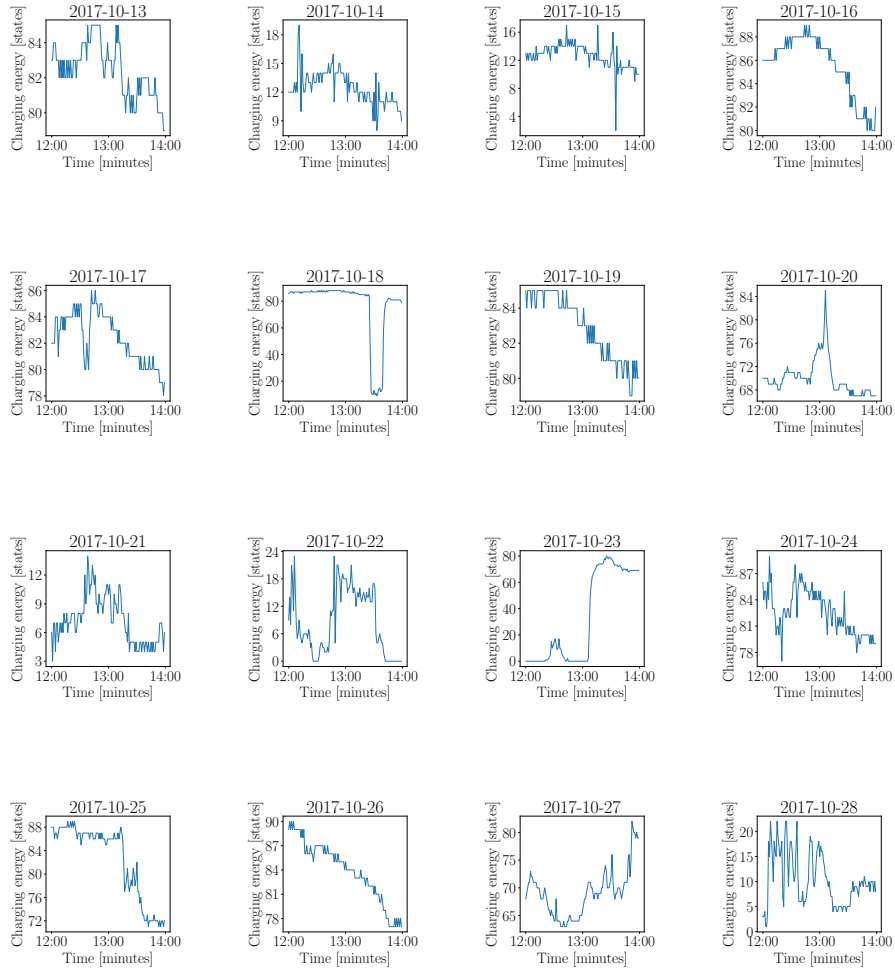
(c) The transition matrix for the mode sub graph including a hysteresis.

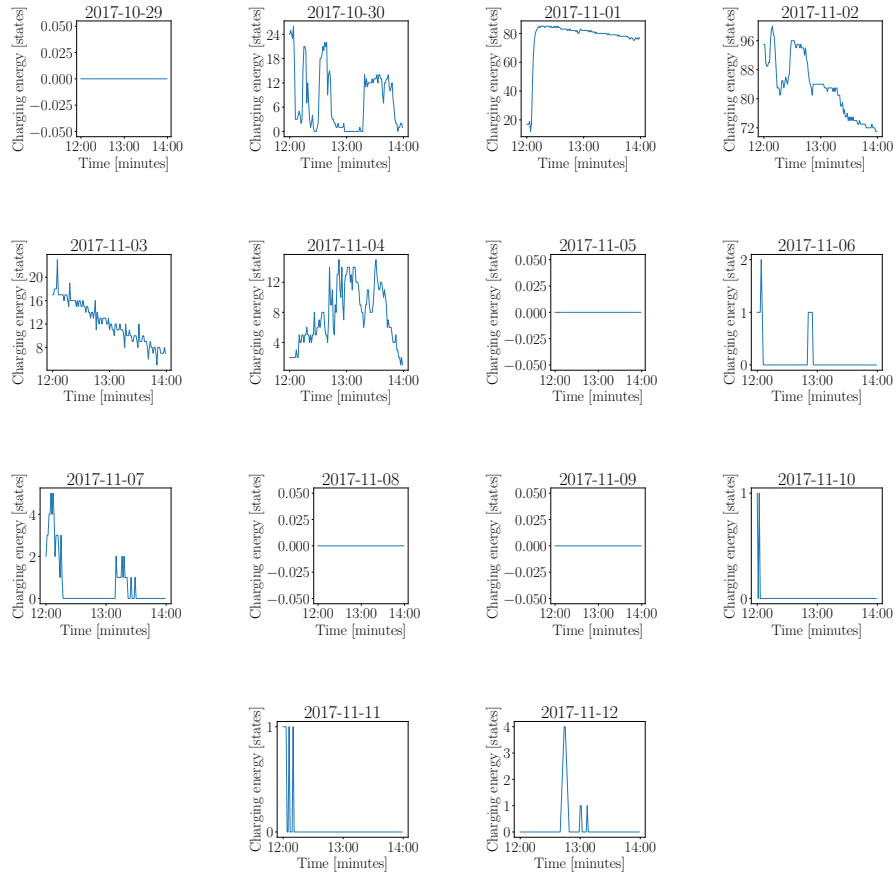
Figure 28: Illustrations for introducing a hysteresis mode behavior with one degraded mode.

A Figures

A.1 Stochastic model and traces: comparison

A.1.1 Trace data





A.1.2 High vs. low traces

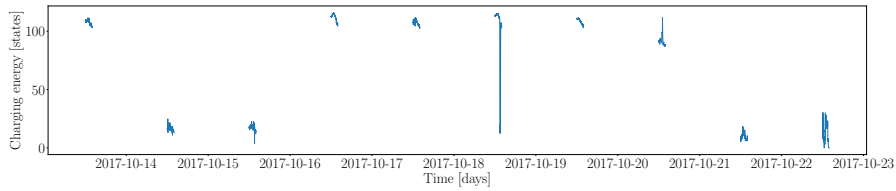


Figure 37: *The 2 hour slices between 12AM and 2PM over 10 days. It can clearly be seen that the days split up into a high and low harvesting group.*

References

- [1] D. Palossi, A. Gomez, S. Draskovic, K. Keller, L. Benini, and L. Thiele, “Self-sustainability in nano unmanned aerial vehicles: A blimp case study,” in *Proceedings of the Computing Frontiers Conference*, pp. 79–88, ACM, 2017.
- [2] B. Buchli, P. Kumar, and L. Thiele, “Optimal power management with guaranteed minimum energy utilization for solar energy harvesting systems,” in *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on*, pp. 147–158, IEEE, 2015.
- [3] P. A. Gagniuc, *Markov Chains: From Theory to Implementation and Experimentation*. John Wiley & Sons, 2017.
- [4] T. P. Baker and A. Shaw, “The cyclic executive model and ada,” *Real-Time Systems*, vol. 1, no. 1, pp. 7–25, 1989.
- [5] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele, “Efficient, long-term logging of rich data sensors using transient sensor nodes,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 1, p. 4, 2017.