A Real-World Evaluation of Energy Budget Estimation Algorithms for Autonomous Long Range IoT Nodes
Philip-Dylan Gleonec, Jeremy Ardouin, Matthieu Gautier, Olivier Berder

To cite this version:

HAL Id: hal-01816578
https://hal.archives-ouvertes.fr/hal-01816578
Submitted on 15 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract—In order to enable IoT nodes to efficiently use their energy harvesting capabilities, algorithms are used to determine a reasonable energy budget and allocate it to node tasks, enabling energy neutral operation. However, most of these algorithms have been implemented and evaluated in simulation frameworks. In this paper, we evaluate the implementation of these algorithms to manage the energy of real-world LoRaWAN IoT nodes. We measure and compare the performance of the different energy budget estimation methods on a commercial LoRaWAN IoT platform. Results show that in this use-case, the choice of algorithm impacts the system Quality of Service by less than 15%. This enables much simpler energy budget estimation methods to be used.

I. INTRODUCTION

Wireless sensor networking has recently been an active research area for both industries and academic institutions. In particular, a lot of work is focused on increasing the battery life of wireless sensor nodes. This can be achieved by reducing the power consumption of individual components, using duty-cycling strategies [1], or using low power transmission technologies such as LoRa. Yet the use of non-rechargeable batteries limits the lifetime of the system and creates waste when this battery is depleted. Thus, the use of energy harvesting and rechargeable batteries has been considered [2]. In such a system, more energy is made available to the node, enabling a longer lifetime of the system.

By adapting the Quality of Service (QoS) of the node to its energy harvesting capabilities, it is possible to reach an energy neutral operation point, and have a theoretically infinite battery life [3]. To do so, energy harvesting nodes use a power manager subsystem. This subsystem can be divided in two elements : an Energy Budget Estimator (EBE) which calculates an energy budget based on the harvested energy and the state of charge of the battery, and an Energy Allocator that decides how to spend this energy budget on different tasks, such as sensing, transmitting a message or processing data. Instinctively, we can see that the node QoS is linked to its EBE : a larger energy budget will enable a higher QoS, but will deplet the battery more quickly.

EBE algorithms can be model-based [4] [5], in which case they use the behavior of the energy source to adapt the energy budget according to an estimation of the future harvested energy. On the contrary, model-free EBE can be used independently of the used energy harvesting source. LQ-tracker [6], Fuzzyman [7] and RLMan [8] are examples of such algorithms, which respectively rely on linear quadratic regression, fuzzy logic and reinforcement learning to calculate an optimal energy budget. The performance of these algorithms has been verified and studied with extensive simulations.

However, to our knowledge, none of these algorithms have been implemented on commercial wireless sensor platforms. Moreover, these designs target short range communications, and the use of energy harvesting for Long Range IoT nodes has not been studied. In this paper, we compare the performance of four different model-free EBE algorithms in the context of a commercial implementation of solar powered LoRaWAN sensor nodes.

The rest of the paper is organized as follows. Section II describes the energy harvesting node platform and our design choices for the power manager. Section III details the implementation and choice of EBE algorithms. Section IV explores the tuning requirements of each algorithm. Section V shows and analyses our experimental results. Finally, Section VI concludes this article.

II. EVALUATION PLATFORM AND REQUIREMENTS

Our evaluation platform targets long range applications with multi-source energy harvesting capabilities [9]. Fig. 1 shows both hardware and software descriptions of the platform. An IoT node is usually composed of a micro-controller, a radio transceiver and some sensors, respectively a STM32L0, a SX1272 LoRa transceiver, and a connector to plug different sensors in our platform. Moreover, three SPV1050 energy harvesters enable simultaneous energy harvesting from a solar panel, a wind turbine and a thermo-electric generator. The harvesters use a common energy storage device which powers the platform through a DC to DC converter. The storage device can be a super-capacitor or a lithium-polymer (LiPo) battery.
In this work, a 7.5 F super-capacitor is used. Due to the board configuration, it is used between 3.7 V and 4.1 V.

The LoRaWAN technology uses the LoRa chirp spread spectrum (CSS) modulation to send short messages over long distances with a limited power consumption. LoRaWAN uses a star topology, where all the nodes send their messages to a gateway over unlicensed frequency bands. These bands are regulated to avoid interferences between radio devices. In Europe, LoRaWAN transmitters must respect a maximum duty cycle (i.e. 1% in the 868-868.6MHz sub-band and 0.1% in the 868.7-869.2 sub-band), which limits the maximal QoS of a LoRa node. The duration of a LoRa transmission depends on the payload size and the CSS spreading factor, which is - in short - the duration of a chirp and is dynamically modified according to the transmission conditions. A higher spreading factor enables a better sensitivity, increasing the transmission range, but reduces the throughput, increasing the energy cost of a transmission.

As we use a different radio technology, the operating conditions of the EBE algorithms differ from the state-of-the-art. The delay between two messages in our application ranges from minutes to hours, while the delay targeted by state-of-the-art EBEs ranges from seconds to minutes. This means that EBE algorithms will need more time to converge to the best result. Moreover, the energy consumption of a LoRa transmission is higher than lower range technologies due to long packet transmissions, which creates higher variations in the battery state-of-charge.

Usually, EBE algorithms are periodically executed, at the end of a timeslot (e.g. 1 hour), and aim to reach the energy neutral operation point for the next timeslot. In our application, to avoid a long convergence time, the EBE algorithm is executed after each transmission. The delay $D[k]$ to the next message is calculated based on the current energy harvesting conditions. This enables greater agility, as the EBE can recover from a previous bad estimation more quickly. However, this also increases the variance of the QoS, as the algorithm has more difficulty converging to an optimized result.

### III. EBE Implementations

The EBE algorithms have been implemented in a C library. Since the micro-controller has no Floating-Point Unit, the library was implemented using a portable fixed-point arithmetics library. This enables the validation of the EBE library on a computer before being implemented on a target node and speeds up development time.

Most EBE algorithms calculate an energy budget $E_B$ for the starting timeslot based on the residual energy $E_R$ of the energy storage device when $E_B$ is calculated [6]. Some EBE algorithms [7] also need the measurement of the harvested energy $E_H$. However, the latter is difficult to perform with a good precision and low energy consumption overhead, and can sometimes be estimated by the residual energy variation $\Delta E_R$. Some algorithms [8] are tightly coupled with the energy allocator and require typical energy consumption $E_{Ctyp}$.

Additionally, we define two applications parameters $D_{max}$ and $D_{min}$, respectively as the maximum and minimum delay in seconds between two transmissions, used to set a minimal QoS. The library uses these parameters to calculate a minimal and maximal energy budget $E_{Bmin}$ and $E_{Bmax}$. We also define $D_{REF}$ as the delay between two transmissions when $E_B$ is equal to $E_{Ctyp}$, set between $D_{min}$ and $D_{max}$. These parameters are dependent on the end-application and are customizable. $D$ (in seconds) is calculated from $E_B$ and $E_{Ctyp}$ as :
\[
D = \frac{D_{\text{REF}} \times E_{\text{Ctyp}}}{E_B}
\]

The library also needs information about the underlying hardware energy capabilities. We denote \(V_{\text{BATT}}, V_{\text{min}}\) and \(V_{\text{max}}\) the current, minimal and maximal battery voltage. \(V_{\text{min}}\) if the voltage for which the platform stops functioning, or can be set with a margin before the battery is fully depleted. When the energy storage is a super-capacitor, its capacitance \(C\) is defined. Based on these parameters, the current, minimal and maximal residual energy in storage \(E_R, E_{R\text{min}}\) and \(E_{R\text{max}}\) are calculated.

The evaluation platform is able to use multiple energy sources simultaneously. Model-based EBE are developed for a single source type, and are unsuitable for this use-case. Thus, model-free EBE algorithms are used. The resulting platform is more generic, and can be used in more end-user applications. In particular, we studied LQ-tracker [6] and Fuzzyman [7], which both showed good results in previous evaluation. LQ-tracker can be implemented using only the energy state-of-charge as input. While Fuzzyman in its original design also requires \(E_H\), which is not measured in our platform. In this work, we consequently use a custom rule-set that uses \(\Delta E_R\) as input instead of \(E_H\), and uses the last energy budget \(E_B[k-1]\) to determine the new \(E_B\). The rules are detailed in Table 1 and only depends on the application and platform parameters and can be reused for a variety of applications and energy sources.

<table>
<thead>
<tr>
<th>(\Delta E_R)</th>
<th>(E_R)</th>
<th>(E_{R\text{min}})</th>
<th>(E_{R\text{max}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt; 0)</td>
<td>(E_{R\text{min}})</td>
<td>(J(E_B[k-1]))</td>
<td></td>
</tr>
<tr>
<td>(= 0)</td>
<td>(J(E_B[k-1]))</td>
<td>(E_{R\text{max}})</td>
<td></td>
</tr>
<tr>
<td>(&gt; 0)</td>
<td>(E_{R\text{max}})</td>
<td>(E_{R\text{max}})</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 1: Fuzzyman custom rule-set.**

In Table 1 \(f(E_B[k-1])\) is a function defined by :

\[
f(E_B[k-1]) = \max\left(\frac{E_{R\text{min}} - E_{R\text{min}}}{E_{R\text{max}} - E_{R\text{min}}}, E_B[k-1]\right)
\]  

Besides, we implemented two naive EBE algorithms : Linear\(_T\) and Linear\(_E\). Linear\(_E\) calculates an energy budget between \(E_{B\text{min}}\) and \(E_{B\text{max}}\) as a prorata of \(V_{\text{BATT}}\) between \(V_{\text{min}}\) and \(V_{\text{max}}\) :

\[
E_B = E_{B\text{min}} + \left(E_{B\text{max}} - E_{B\text{min}}\right) \times \left(\frac{V_{\text{BATT}}}{V_{\text{max}}} - \frac{V_{\text{min}}}{V_{\text{max}}}\right)
\]

Linear\(_T\) calculates a fixed delay between \(D_{\text{min}}\) and \(D_{\text{max}}\) as a proportion of \(V_{\text{BATT}}[k]\) between \(V_{\text{min}}\) and \(V_{\text{max}}\). This delay is then converted to an energy budget based on \(E_{\text{Ctyp}}\). The full equation is given by :

\[
E_B = D_{\text{REF}} \times \frac{E_{\text{Ctyp}}}{D_{\text{max}} - (D_{\text{max}} - D_{\text{min}}) \times \left(\frac{V_{\text{BATT}}}{V_{\text{max}} - V_{\text{min}}}\right)}
\]

These two algorithms are used as a performance baseline to evaluate the advantages of more advanced control algorithms. Their advantage is their simplicity, as no settings are required beyond the common platform settings. Finally, to avoid sharp \(E_B\) variations due to temporary change in energy harvesting conditions, all algorithms outputs are smoothed by an adaptive filter with a parameter \(\beta = 0.9\) :

\[
E_{B\text{filtered}}[k] = \beta \times E_B + (1 - \beta) \times E_{B\text{filtered}}[k-1]
\]

**IV. EBE ALGORITHMS TUNING**

All the selected algorithms can be tuned in order to improve their performance. The library has been designed to be portable, which enables the algorithms to be tuned in a simulation framework using the same code and same results. Indeed, it is much longer to run a full measurement campaign to validate an algorithm tuning settings than to run a simulation. This is particularly true for algorithms which provide many settings, such as Fuzzyman.

Each EBE algorithm has its tuning capabilities. Linear\(_T\) and Linear\(_E\) only need the common platform parameters, \(V_{\text{min}}\) and \(V_{\text{max}}\), which can be adjusted to increase the efficiency of the EBE. LQ-tracker requires more parameters : it takes as input an energy target level \(E_{\text{TARGET}}\), a step-size \(\mu\) and two parameter vectors \(\Phi\) and \(\Theta\). \(E_{\text{TARGET}}\) value is dependent on the application. \(\mu\) is set by default to 0.001 and impacts how quick LQ-tracker converges. Fuzzyman is the hardest one to tune, as it can be entirely redesigned to fit the application. Indeed, it requires four thresholds \(THR_{\text{Empty}}, THR_{\text{Full}}, THR_{\text{Positive}}\) and \(THR_{\text{Negative}}\) to set the limits of the fuzzy inputs. Moreover, the entire rule-set may be redesigned to provide better performances.

For our simulation, the delay between two messages is set between of 15 minutes to 3 hours. The energy consumed by a LoRaWAN transmission is estimated to 140mJ with a SF7 spreading factor. We set \(V_{\text{min}}\) and \(V_{\text{max}}\) respectively to 3.7 V and 4.1 V, as for our real energy storage. The system was tested with a solar energy source, using EnHANTS [10] measurement data. The simulated solar panel measures 3.2 cm per 4.2 cm with an efficiency of \(\phi = 10\%\), like the one used in experimentation. The simulation results, without tuning, are shown in Table II. The mean value \(\bar{x}\) and standard deviation \(\sigma\) of \(V_{\text{BATT}}, E_B\) and \(D\) are presented.

We can first notice that all algorithms have some failed transmission, meaning some transmissions were aborted due to \(V_{\text{BATT}}\) being lower than \(V_{\text{min}}\). Most of them occur around a lack of data in the solar traces, which we kept to simulate a long period without energy. Two algorithms, Fuzzyman and Linear\(_T\), are more conservative, which can be seen as they keep a higher and more stable \(V_{\text{BATT}}\), and thus emit less messages than the other two. \(E_B\) of Fuzzyman is high and unstable, which shows that it sometimes tries to deliver an \(E_B\) too high and can not stabilize. Linear\(_E\) and LQ-tracker, on the other hand, take more risks, showing a lower mean \(V_{\text{BATT}}\), and are rewarded with a higher number of transmissions.
In Table III, while the fuzzification thresholds $\phi \in [6]$. Fuzzyman was given a new ruleset, shown to their default $\Theta$ of LQ-tracker to 0.005, and leave $\phi$ when it is sufficient. The energy storage is nearly depleted and take more risks if the environment conditions are bad for an extended time period. In the evaluated algorithms, LQ-tracker seems to be the most conservative, starting to reduce the delivered $E_B$ when the state of charge goes under 70%. The Linear algorithms and Fuzzyman are more aggressive, delivering maximum $E_B$ when the state of charge is respectively over 35% and 17%.

V. REAL-WORLD EBE MEASUREMENTS

During our experiments, each hardware platform is powered by a solar panel which can provide up to 5 V and 40 mA from direct sunlight. The experiment was run for seven days in an office settings. Each algorithm ran on a different hardware platform. All sensors were located next to a north oriented window, and ran in parallel to reduce differences in ambient energy. After each transmission, the energy budget and delay to the next transmission are calculated and sent through an UART, along with $V_{BATT}$ and a count of successful transmissions. This UART transmission is logged by a computer, parsed and analyzed to produce the graphs shown in Fig.3.

The number of successful transmissions is approximately the same for all algorithms, within a 15% margin. Only one transmission failed during the fifth night for Fuzzyman. This shows the algorithms are quantitatively equivalent provided they are properly tuned, which confirm the results obtained through simulation. Qualitative differences can however be observed between the different EBE implementations.

As expected, LQ-tracker, by being more conservative, keeps its $V_{BATT}$ higher than the other algorithm, which makes it more able to survive long periods without energy. On the contrary, Fuzzyman delivers a competitive number of transmissions but keeps its $V_{BATT}$ much closer to the failure threshold, making it more sensible to lack of energy input, as seen with the failed transmission. As Fuzzyman is highly tunable, this could be optimized by a carefully designed ruleset, to provide characteristics closer to LQ-tracker.

Moreover, Fuzzyman and Linear$_T$ show a high variance in the delivered $E_B$ and delay $D$, which can be explained by the quick decrease of $E_B$ when $V_{BATT}$ goes under a certain threshold, defined by the tuning of the algorithm. This quick $E_B$ decrease translates into a much longer delay $D$, and decreases the performances of the algorithm, despite the aggressive tuning. On the other hand, Linear$_E$ and LQ-tracker reduce $E_B$ more progressively, which translates to a lower mean $D$ and improved performances. This behavior also depends on how $D$ is calculated from $E_B$, and could be varied using a different energy allocation method.

<table>
<thead>
<tr>
<th>TABLE II: Simulation before algorithm tuning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{BATT}$ (V)</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>4.060</td>
</tr>
<tr>
<td>3.988</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III: Fuzzyman new ruleset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta E_B$</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>$&lt; 0$</td>
</tr>
<tr>
<td>$\geq 0$</td>
</tr>
<tr>
<td>$&gt; 0$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE IV: Simulation after algorithm tuning.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{BATT}$ (V)</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>4.060</td>
</tr>
<tr>
<td>3.988</td>
</tr>
</tbody>
</table>

![Fig. 2: $E_B$ as a function of the State-of-Charge.](image-url)
Finally, it can be observed that all algorithms show a sharp rise of $E_B$ when the energy storage gets replenished. As all the evaluated algorithms are model-free, they have no means to predict an approximate future energy state of charge. In our use-case, this is typically the case in the morning: the calculated $D$ is long at the end of the night, when the energy storage starts getting replenished. When this long delay $D$ is elapsed, the storage has been charging for a long time, and the EBE can deliver a higher $E_B$. This effect would be less important with a model-based EBE algorithm, as it would predict that energy was going to be harvested.

VI. CONCLUSION

This paper presents a practical implementation of Energy Budget Estimator algorithms for LoRaWAN IoT nodes and their optimization. We show that the choice of network technology has an impact on EBE algorithms implementation and performance. The use of a simulator shows that in our use-case, there are few differences between the different algorithms, provided they are all properly tuned. This enables the use of simpler EBE algorithms, which can then be more easily integrated in mathematical framework to size energy storage and energy sources based on Quality-of-Service requirements. Future work will focus on the development of such a framework, and on the allocation of the energy budget, especially for multi-task contexts.

REFERENCES


