Intermittent Low-Power Wide Area Networks

Charalampos Orfanidis
Technical University of Denmark
Kgs. Lyngby, Denmark
chaorf@dtu.dk

Adam Ømosegård Bischoff
Technical University of Denmark
Kgs. Lyngby, Denmark
adam@bischoff.dk

Luca Pezzarossa
Technical University of Denmark
Kgs. Lyngby, Denmark
lpez@dtu.dk

ABSTRACT

Low-Power Wide Area Networks (LPWAN) offer long-range communication with low energy consumption, making them ideal for IoT applications powered by energy harvesting. However, unpredictable energy harvesting rates can lead to sub-optimal device operation. To tackle this, the intermittent computing paradigm has been proposed. In this paper, we explore the combination of intermittent computing and LPWANs by proposing four different communication solutions based on LoRa and designed to match common application scenarios and requirements. For the solutions, we also present experimental energy estimation models, which we evaluate against measurements.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; • **Networks** → **Mobile networks**.

KEYWORDS

LPWAN, Battery-free, IoT

ACM Reference Format:
Charalampos Orfanidis, Adam Ømosegård Bischoff, and Luca Pezzarossa. 2023. Intermittent Low-Power Wide Area Networks. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3570361.3615729

1 INTRODUCTION

The use of Low-Power Wide Area Networks (LPWAN) in the Internet of Things (IoT) ecosystem offers long-range communication in low energy, which is a necessary feature in new application scenarios such as smart cities, smart agriculture, and many more. The number of IoT devices including LPWANs is continuously increasing, with the estimated number of devices already in the billions [2]. Powering these devices has become a challenge: it is unsustainable for the environment to use batteries and, in some cases, replacing batteries is very difficult due to the position of the devices (e.g., smart wall applications with nodes inside the wall).

An approach to address this challenge is to supply energy to IoT devices through energy harvesting sources. However, the energy harvesting rate is often unpredictable. An IoT device solely dependent on energy harvesting might operate sub-optimally or turn off completely because of insufficient energy. To tackle this, intermittent computing [6] has been proposed, where a program is only executed periodically, separated by sleep-state periods where the device uses little energy or is turned off completely. For example, the harvested power can be stored in a capacitor and, before the energy runs out, the system state is saved to non-volatile memory to continue when there is enough energy again. Intermittent computing has been utilized for several computing systems [3, 4] as well as for wireless communication systems using Bluetooth [5].

Applying the same approach to power LPWANs nodes is challenging. For instance, LoRa [1] might have long transmission times to transmit a single packet (sometimes more than 1.5 s) and variable consumption depending on the application. If the stored energy is not enough to transmit the whole packet, the received data can be corrupted and the packet might need retransmission.

To explore the combination of intermittent computing and LPWANs, we present four different communication solutions based on LoRa and designed to match common application scenarios and requirements. The distinctions between the four solutions are in two key areas. The first distinction is in whether the program is able to reliably know the energy *budget* to estimate how many bytes it can afford to transmit without reaching energy depletion. This also assumes an accurate estimation of the program consumption. The second distinction is whether the available communication channel is *simplex* or *duplex*. In the *duplex* scenario, the receiver can acknowledge the correct data reception. In addition, we also present experimental models to estimate the energy

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3615729

consumption for the solutions, which we evaluate against measurements.

2 PROPOSED SOLUTIONS

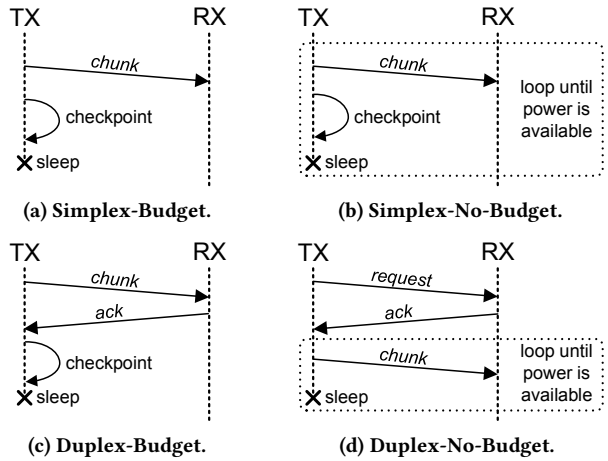
The overall functionality of all four proposed solutions is the same: transmitting a potentially large amount of data (referred to as *message*) over a series of smaller segments (referred to as *chunks*) in the context of intermittently powered devices.

For all solutions, transmission is carried out as a sequence of *wake-up cycles*, where the device wakes up, obtains the data to send, checkpoint the data sent in a non-volatile Flash memory, and enters the sleeping phase. At first, the device wakes up (i.e. when a certain voltage is reached on a supercapacitor, a predefined time has elapsed, or due to an external trigger). Then, it reads a checkpoint in its non-volatile storage. If the checkpoint reports that a *message* is present and not yet transmitted completely from a previous *wake-up cycle*, its data is used. If not, new data might be generated depending on the use case (e.g., reading from sensors or acquiring a photo from a camera) and saved to the non-volatile memory. Some or all of the data is then transmitted using LoRa and a new checkpoint is made to mark what data has been transmitted ensuring that the same data is not retransmitted in the next *wake-up cycle*. Afterward, the device enters the sleeping phase.

Solution 1: Simplex-Budget. In this version the device knows the available *budget* and starts a *chunck* transmission immediately. On completion, it checkpoints how much of the *message* was transmitted into the non-volatile memory to resume transmission accordingly in the next *wake-up cycle*, as shown in Figure 1a.

Solution 2: Simplex-No-Budget. In this solution, the *budget* is unknown. Thus, the device splits the *message* up into small *chunks*, which are sent sequentially during the same *wake-up cycle*. Without knowing the available energy, the transmission might get interrupted at any point. Therefore, the device checkpoints after every *chunk* transmission, as shown in Figure 1b. To avoid corruption of the checkpoint caused by sudden power-offs while writing, a double buffer is used. During the beginning of a device's *wake-up cycle*, the latest valid DB checkpoint is read and transmission of the *message* continues accordingly.

Solution 3: Duplex-Budget. In this solution, when the device has transmitted data in the same fashion as Solution 1, instead of going to sleep, it begins to listen for a response from the receiver, as shown in Figure 1c. If the data was correctly received (*chunks* are identified through a unique ID) the device will create a new checkpoint, causing it to transmit the next *chunk* of data in the next *wake-up cycle*. If no acknowledgment is received or it does not match the expected ID, the device will instead retransmit the same *chunk*. Knowing the *budget* allows for leaving enough margin after transmission for the acknowledgment reception.



(a) Simplex-Budget. (b) Simplex-No-Budget. (c) Duplex-Budget. (d) Duplex-No-Budget.

Figure 1: *Wake-up cycle* sequence diagrams for the four proposed solutions.

Solution 4: Duplex-No-Budget. This solution requires no assumptions about its energy *budget* for sending a packet while also enabling data to be sent reliably. The traditional approach to acknowledgments is to send it immediately upon receiving some data, i.e. in the end of the *wake-up cycle*. Instead, in our solution the acknowledgment has been moved to the beginning of the next *wake-up cycle* by first transmitting a short *request* packet upon wake-up, as shown in Figure 1d. The *request* acts as an announcement to the receiver that the device has woken up and is ready to receive its acknowledgment. The receiver then responds with an acknowledgment, reporting which bytes are yet to be received. The remaining data is then transmitted in *chunks* until either the stored energy is depleted or the *message* is completely sent. Moving the acknowledgment to the beginning of the *wake-up cycle* is essential as this solution has no way of estimating when to stop transmitting packets and instead listen for an acknowledgment.

3 PRELIMINARY EVALUATION

All the results are obtained with an experimental setup consisting of the ESP32 microcontroller by Espressif and a LoRa module based on the RF96 transceiver by HopeRF. The setup is the same on both the transmitter and receiver sides. On the transmitter side, power consumption is sampled every 10ms using an Analog Discovery 2 virtual oscilloscope. The used LoRa settings are: spreading factor: 7, bandwidth: 125 kHz, coding rate: 4/5, frequency: 868 MHz, CRC enabled, explicit header mode.

Table 1: Overview of the energy estimation in Joules of each segment, where m is the *message* size in bytes, c is the *chunk* size in bytes, and p is the listening duration in milliseconds.

Segment	Energy estimation (J)
Boot microcontroller, initialize non-volatile memory and LoRa module, and enter sleep	0.07615
Store the *message* in non-volatile storage	$5.564 \cdot 10^{-6}m + 0.004533$
Read the *message* from non-volatile storage	$6.154 \cdot 10^{-7}m + 0.001564$
Transmit a LoRa packet	$1.126 \cdot 10^{-3}c + 0.02248$
Listen and receive for a LoRa packet	$2.364 \cdot 10^{-4}p + 0.006622$
Store checkpoint data in the non-volatile memory	0.006944
Transmit *chunks* while storing checkpoints using a double buffer	$0.001534c + 0.1211$
Read checkpoint from the double buffer	0.01326

The execution flow of the 4 different solutions can be divided into the segments listed in Table 1. For each of these segments, we conducted a series of experiments and used regression analysis to derive a formula to estimate energy consumption as a function of the *message* size, the *chunk* size, and the listening duration. These estimations can be combined according to the execution flow for the four proposed solutions to enable the prediction of the total energy cost of each approach. In principle, this can be done for any use case desired that can be represented as a combination of these segments. The same energy consumption characterization can be carried out for different microcontrollers and transceivers.

To compare between solutions, we transmit an image of 27254 B assuming the employment of a supercapacitor as the energy storage unit. Figure 2a shows the estimated energy required to transmit the image for the four solutions assuming capacitor sizes from 0.2 to 3 F. We can observe that using small capacitor sizes increases the overall energy needed to transfer the image due to the overhead consumption of having multiple short *wake-up cycles*. Moreover, *No-Budget* solutions seem to consume more energy than the *Budget* ones. However, for real applications, the *No-Budget* solutions allow for better utilization of the available energy compared to the *Budget* ones, where a conservative estimation made in advance will often lead to available energy left at the end of a *wake-up cycle*, which could have been used for transmitting a larger *chunk*. The *No-budget* solutions work well for applications where energy estimations are difficult or limited, and utilizing the entire energy buffer becomes important.

Figure 2b shows the measured energy of the full *message* transmission compared with the estimated one. An interesting observation is that for this use case, the *Duplex-No-Budget* solution consumes less energy than the *Simplex-No-Budget* one. This can be explained by the fact that once a *chunk* has been transmitted and acknowledged by the server for *Duplex-No-Budget*, it does not need to be retransmitted. On

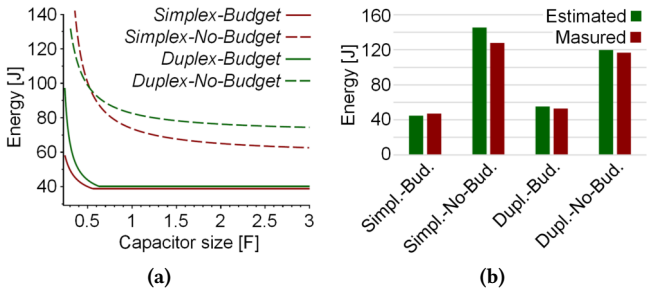


Figure 2: (a) Estimated energy to transmit a 27 kB image against different capacitor sizes and (b) comparison between estimated and measured energy.

the contrary, for the *Simplex-No-Budget* solution, if a *chunk* is transmitted but not yet checkpointed, it has to be transmitted again. Overall, the measured energy consumption are very close to the estimated ones, with differences from a minimum of 3% to a maximum of 14%.

REFERENCES

[1] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. Understanding the limits of lorawan. *IEEE Communications magazine*, 55(9):34–40, 2017.

[2] R. Ahmad, M. A. Asim, S. Z. Khan, and B. Singh. Green iot—issues and challenges. In *Proceedings of 2nd international conference on advanced computing and software engineering (ICACSE)*, 2019.

[3] S. Ahmed, N. A. Bhatti, M. H. Alizai, J. H. Siddiqui, and L. Mottola. Efficient intermittent computing with differential checkpointing. In *Proceedings of the 20th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, 2019.

[4] J. De Winkel, V. Kortbeek, J. Hester, and P. Pawełczak. Battery-free game boy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(3):1–34, 2020.

[5] J. De Winkel, H. Tang, and P. Pawełczak. Intermittently-powered bluetooth that works. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, 2022.

[6] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel. Intermittent computing: Challenges and opportunities. *2nd Summit on Advances in Programming Languages (SNAPL 2017)*, 2017.