MARS: MAximizing throughput for MPPT-based self-sustaining LoRa Systems

Ruben Dominguez
Texas A&M University-Corpus Christi
Corpus Christi, TX, USA
rdominguez6@islander.tamucc.edu

Jose Baca
Texas A&M University-Corpus Christi
Corpus Christi, TX, USA
jose.baca@tamucc.edu

Chen Pan
University of Texas at San Antonio
San Antonio, TX, USA
chen.pan@utsa.edu

ABSTRACT

Due to the insufficient transient amount of energy supplied from ambient energy sources and constrained amount of energy storage in super-capacitors, energy harvesting (EH) nodes are limited with operations and vulnerable to frequent faults due to energy scarcity. Consequently, such faults will reduce reliability and energy utility due to data collisions, lost data, or idle listening. To address these challenges, this work implements a novelty task scheduling scheme to minimize energy waste and maximize throughput under these scenarios and constraints. To demonstrate the effectiveness, we use a green test bed using LoRa nodes for evaluation.

CCS CONCEPTS

• **Computer systems organization → Embedded software**.

KEYWORDS

Self-sustaining, Energy Harvesting, Battery-less, Green IoT, LoRa

ACM Reference Format:

Ruben Dominguez, Jose Baca, and Chen Pan. 2024. MARS: MAximizing throughput for MPPT-based self-sustaining LoRa Systems. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649476.3658722

1 INTRODUCTION

Energy harvesting (EH) has been increasingly employed in the modern IoT ecosystem due to its promising low cost, pervasive availability, and environmental friendliness features. By harvesting and harnessing energy from ambient sources such as solar, thermal, wind, and kinetic energy [11, 13], IoT devices can be self-sustained in computing and communicating for long-term deployment.

However, due to the weak and unstable nature of ambient energy sources [5], EH nodes are vulnerable to frequent power outages resulting in the data corruption of ongoing communication. What is even worse is that such a drastic voltage fluctuation can yield the reset of the volatile time module onboard causing synchronization problems between the receiver and the sender.

To initiate communication, a synchronized timeline is required where both the sender and receiver need to turn on their radio simultaneously. While there should be no problem with conventional battery or main electricity-powered IoT systems where the receivers are always listening to the request-to-send (RTS) beacon, it is inapplicable under energy harvesting environments. Due to limited power supply under energy harvesting environments, self-sustaining IoT devices will only turn on the radio intermittently based on the communication schedule to maintain sustainability and ensure successful communication. However, After resetting the time module due to power interruption, the individual node will not be able to turn on its radio at the correct schedule. As such, time and energy efforts are required for two devices to synchronize and start transmitting data. To mitigate the occurrence of desynchronization issues, existing research targets reducing energy consumption and improving energy efficiency through optimization of routing, scheduling, resource allocation, etc.[2, 6–8, 10, 16, 17].

To maximize the throughput of IoT networks with limited and unstable power in energy harvesting scenarios, it's important to integrate energy-efficient EH node task scheduling schemes to ensure that both the senders and receiver are aware of their dedicated time slots for communication to avoid unnecessary interference and collision. Besides, by integrating existing anti-collision and fault-tolerant techniques nodes can not only retain synchronization but also minimize energy consumption during the communication. Therefore, instead of merely increasing reliability, this work will enable an energy-efficient task scheduling design for EH IoT devices to maximize the individual throughput for all of the nodes in a network after a sufficient amount of energy has been stored in the on-board super-capacitors for efficient implementation of the featured on-board MPPT. The proposed framework will upgrade the adopted healing techniques and resource allocation methods in [3] for enabling fault tolerance and anti-collision. In this way, nodes can retain synchronization as well as minimize energy waste.

To demonstrate the effectiveness, we will take the increasingly popular Long-range Wide Area Network (LoRaWAN) protocol as the backbone for the upgrade. Through a hybridization of MPPT-based hardware with energy-efficient task scheduling, we will enable effective use of maximum power of harvested energy as well as maximize communication under energy harvesting scenarios. To the best of our knowledge, this is the first work to propose a set of energy-efficient techniques that addresses all angles of energy waste with increased efforts of throughput under energy harvesting scenarios. The techniques included can be easily transferred to upgrade other MAC protocols for enabling effective throughput increase under energy harvesting scenarios. In a nutshell, the contribution of this work is as follows.

1) Once the super-capacitor is adequately charged, anti-collision protocols will activate during initialization. This involves

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658722

uniquely assigning LoRa communication resources—frequency channels, spreading factors, and time division scheduling.
2) A deadlock-break technique is used after a repetition to minimize energy loss due to collisions during the initialization.
3) Fault tolerance is enabled for EH nodes experiencing power outage, acknowledging and re-synchronizing themselves to the network during runtime. This allows them to resume scheduled communication without impacting other nodes.
4) The lost/unacknowledged packets will be retained by each individual and retransmitted during the upcoming scheduled communication to improve the throughput.

The remainder of this paper is organized as follows. Section 2 contains works relative to the considered EH system. Section 3 conducts system modeling which describes the relationship between the major considerable factors of the network. After that, Section 4 formulates the optimal task scheduling in the duty cycle and proposes the corresponding minimal energy waste algorithms. Section 5 conducts experimentation to demonstrate the performance of the self-sustained and energy-efficient algorithms. Finally, Section 6 concludes this work.

2 RELATED WORK

In designing practical self-sustaining EH networks, special focus has been made on network topology and energy consumption [5] in two angles, the network layer attributes for duty cycle scheduling and routing optimization techniques, and the data link layer attributes for optimal resource allocation and adaptation to physical layer issues in EH systems. Since power outage is frequent under realistic EH scenarios, conventional methods and policies are rendered not practical for EH nodes [12].

In an effort of making designs more suitable to EH scenarios, utilization in new hardware focused on energy conservation must incorporate safety mechanism techniques in designing to address not just the synchronization issues that arise from utilization of sleep technology but also maximizing operations effectively. Sleep-wake policies with wake-up radio technology are made in works such as [9], proposing an extension of a previous throughput optimization policy and resulting in an increase in throughput as well as stability in the data queue. A similar wake-up policy is proposed in [2], proposing an opportunistic MAC protocol to lessen the fluctuations of quality of service between different day times through a new energy manager, but unfortunately lacks any sort of acknowledgment in the intermittent behavior of EH nodes or any attributing mechanisms for addressing the practically frequent power outage. Though these works provide energy-efficient benefits to increase reliability in sustainability, they still either lack MAC layer attributions in addressing the physical problems or if including a MAC layer attribute fail in acknowledging the practicality of power outage at all.

There are several papers that attempt to further diversify the number of solutions applied for EH systems [10]-[16]. In [10], a hybrid physical and MAC layer approach using passive wake-up radio technology is proposed to minimize idle listening based on previous works in the literature. [17], employs a hybrid physical and network layer-focused solution for duty cycle scheduling with presented algorithms with proven performance in prolonging node life. [16] approaches a triple hybrid physical, network, and data link layer solution proposing a resource allocation scheme jointly with a duty cycle alternating optimization scheme targeting higher throughput and energy efficiency with outperforming results. In [3] a fault tolerant resource allocation scheme is proposed for power outage synchronization, but lack efforts for maximizing throughput.

While all the mentioned techniques provide unique benefit to overall energy consumption and broaden approaches in energy saving schemes for EH nodes, they still fall short of practical implementation due to a lack of attributing MAC layer solutions in addressing the synchronization problems that are practically frequent under energy harvesting scenarios. Most works that do contain hybrid energy efficient solutions with necessary fault tolerant schemes to address these problems, lack throughput optimization efforts.

Therefore, this paper aims to broaden more direction for self-sustained EH networks with fault tolerant synchronization strategies while jointly handling scheduled routine communication with the synchronized nodes in the system, and also develop an energy waste reduced task scheduling on EH nodes to maximize efficiency.

3 SYSTEM MODEL AND ANALYSIS

In this section, systemic modeling including network modeling, an energy model, and task scheduling analysis will be conducted to facilitate the hybridization between MPPT-based hardware and energy-efficient task scheduling that enables effective use of maximum power of harvested energy as well as maximizing communication under energy harvesting scenarios.

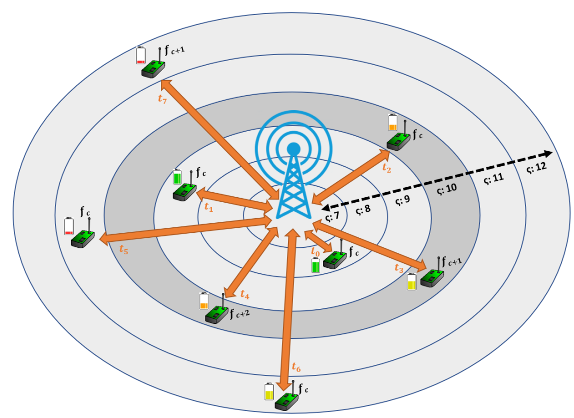

Figure 1: System Model

3.1 Network Modeling

In Fig. 1, a central gateway LoRa receiver is positioned at the center of a defined region $\mathcal{A} \subset \mathbb{R}^2$. This area contains N LoRa EH transmitter nodes, randomly deployed and unknown to the gateway. The gateway connects to a local LoRa server via a back-haul link and is adequately powered. The EH LoRa nodes rely on harvested ambient energy for all runtime operations—sensing, processing, and transmissions. Each EH node initializes with a unique combination of resource allocations, this includes frequency channel f_c, spreading factor ς, and sleep time τ_s. Following initialization, all N EH nodes are allotted independent scheduled time slots to communicate with the gateway.

In LoRaWAN, the gateway assigns the spreading factor ς based on the received RSSI of the node compared to receiver sensitivities. These sensitivities, $S[4]$, are calculated using the formula obtained from the Semtech LoRa radio chip supplier [15]. The parameters f_c and ς self-configure in LoRa radio chips during runtime without gateway assistance. Additionally, the gateway calculates a sleep time τ_s unique to each node during initialization, based on the initialization duration and the number of nodes allocated resources. This ensures each node in the system has an independent scheduled time for communication with the gateway after sustained sensing post-initialization.

3.2 Energy Modeling

Ambient sources charge the on-board super-capacitor to sufficient charging E_S before initial booting. From the initial power-up at time t_0 until the gateway connection for initialization at t_i, EH nodes efficiently harvest maximum energy using continuous harvesting power $P_h(t)$ via MPPT.

Upon booting, each node requests network joining for initialization, incurring energy consumption E_R. This resource allocation request involves energy consumption E_b for transmission and brief power consumption for listening $P_l(t)$ to await a gateway acknowledgment over a time period τ_a, determined by the dynamically allocated resources. Concurrently, energy is consumed by the active state core with active power $P_a(t)$ during the node's ping and acknowledgment listening, equating to twice τ_a (Equation 1).

$$E_R = E_b + \int^{\tau_a} P_l(t)\,dt + 2\int^{\tau_a} P_a(t)\,dt \tag{1}$$

The energy consumed from requesting before being acknowledged by the gateway can be a probabilistic k amount of times, due to the stochastic order for node resource allocation during the initialization. Given the sufficient charge E_S at t_0 in the super-capacitor, the residual energy $E_r(t_i)$ of EH nodes after being successfully initialized at t_i can be expressed as shown in Equation 2.

$$E_r(t_i) = E_S + \int_{t_0}^{t_i} P_h(t)\,dt - \sum_{n=1}^{k}(E_R) \tag{2}$$

Here, the total usable residual energy $E_r(t_i)$ at any time after initialization is the sum of the charged super-capacitor E_S and the harvested energy from t_0 to t_i, minus the energy cost incurred during the requesting process.

After successful initialization and resource allocation, the gateway instructs nodes to sleep for duration τ_s as scheduled in the duty cycle. During this time, nodes consume significantly less power $P_s(t)$ than during their active state, operating on the ultra-low-power core while in deep sleep. When the scheduled time arrives for nodes to communicate within the duty cycle, they attempt to send previously sensed information, followed by listening for an acknowledgment. If acknowledgment is not received within time τ_a, the nodes store the sensed information and repeat this process up to k times, provided there is sufficient energy. The energy consumed during these attempts E_A, where $E_{s,b}$ represents the energy used for data sensing and communication, is shown in Equation 3.

$$E_A = E_{s,b} + \int^{\tau_a} P_l(t)\,dt + 2\int^{\tau_a} P_a(t)\,dt \tag{3}$$

Finally, based on the energy during initialization $E_r(t_i)$, Equation 4 expresses the residual energy $E_r(t_s)$ for adequate communication of any post-initialization instances in time t_s.

$$E_r(t_s) = E_r(t_i) + \int_{t_i}^{t_s} P_h(t)\,dt - \sum_{n=1}^{k}(E_A) \tag{4}$$

3.3 Energy-waste Reduction Analysis

To optimize energy efficiency in energy harvesting systems with super-capacitor storage, efficient energy usage during initialization and runtime communication are crucial due to the energy scarcity from ambient sources. While employing specific methods for minimize energy wastage, such as collision avoidance through resource allocation and initial scheduling during duty cycle initialization [3], it becomes essential to reduce idle listening while enhancing data integrity. Figure 2 demonstrates strategies like use of on-board data storage and promptly attempting sending the concatenated data when sufficient energy is available. These techniques ensure the successful transmission of both new packets and any previously unacknowledged packets.

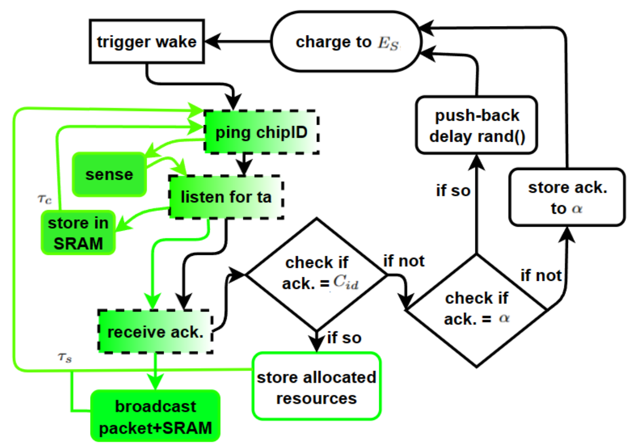

Figure 2: EH node flowchart

4 ENERGY WASTE REDUCTION ALGORITHMS

In this section, we will propose an energy-waste-reduced task scheduling for energy-harvesting powered LoRa nodes based on the model and analysis from Section 3, the design will be conducted during runtimes for the gateway and EH nodes, since as mentioned previously in Section 3.3, methods for resource allocation and initial scheduling during initialization were adopted from [3].

4.1 Gateway

As shown in Algorithm 1, the gateway will hop through spreading factors and frequency channels where nodes are known to be allocated in memory to collect data given the initial inputs: the number of nodes in the network N, the vector set of parameters for each node stored $p\langle C_{id}, \varsigma, f_c, \tau_s\rangle$, the initialization/healing frequency f_i, the tracking list for the number of nodes stored in each spreading factor $n_\varsigma[4]$, the frequency channels $f_c[8]$, the time on air τ_a, and the healing time set τ_h. Hence, the outputs include the stored parameters for every node in the system $p\langle C_{id}, \varsigma, f_c\rangle$, the tracking the number of nodes allocated in each spreading factor, $n_{\varsigma}[4]$, and directed sleep times τ_s.

Algorithm 1: Run-time - Gateway

Input: N, f_i, $p\langle C_{id}, \varsigma, f_c, \tau_s\rangle$, f_i, $n_{\varsigma}[4]$, $f_c[8]$, τ_a, τ_h
Output: $p\langle C_{id}, \varsigma, f_c\rangle$, $n_{\varsigma}[4]$, τ_a, τ_h

while *ON* **do**
  for *i = 1 to 4* **do**
    if $n_{\varsigma}[i]$ *!= 0* **then**
      for *j = 1 to* $n_{\varsigma}[i]$ **do**
        listen on $f_c[j]$ at $n_{\varsigma}[i]$ for τ_a
        if *received packet* **then**
          transmit received ack. + τ_s
        end
        else
          listen on f_i for τ_h
          if *received packet* **then**
            for *l = 1 to N* **do**
              if $p_n^{C_{id}}[l]$ *!=* C_{id} **then**
                $C_{id}check{+}{+}$
              end
            end
            if $C_{id}check == \sum_{k=1}^{4}(n_{\varsigma}[k])$ **then**
              for *l = 1 to N* **do**
                if $p_n^{C_{id}}[l] == C_{id}$ **then**
                  calculate τ_s based on reserved schedule
                  with C_{id} header, transmit $p[l]$ & τ_s
                  $C_{id}check \leftarrow 0$
                  $j{+}{+}$
                end
              end
            end
          end
        end
      end
    end
  end
end

While the gateway remains ON in a server it will for every spreading factor check if there are any nodes allocated a channel on that spreading factor $n_{\varsigma}[i]$, and if so, will listen for each packet frequency hopping through $f_c[j]$ to receive scheduled packets followed with sending acknowledgments with directed sleep time τ_s.

If after time on air of a LoRa ping τ_a the gateway fails to receive a packet, it will listen on the initialization frequency f_i for healtime t_h which is tuned to recover nodes requesting to join the network. If a node requests, the gateway checks in local storage for the previously allocated parameters $p[l]$ during the initialization along with new sleep time τ_s and continues duty cycle as scheduled.

Notice if the gateway didn't find the desynchronized node during the initial healing, the gateway will continue the routine communication schedule and try again during the next healing cycle until the gateway does receive that request from the desynchronized node, not interfering the rest of the nodes scheduled in duty cycle.

4.2 Run-time EH Node

During the EH node scheduled runtime, as shown in Algorithm 2, the initial inputs given is its local chipID C_{id}, the initially allocated resources $p\langle\varsigma, f_c\rangle$, the charge time τ_c, the time on air for a packet τ_a, and sleep time τ_s. The EH node will wake up either from brief charge time τ_c or from directed sleep time τ_s to listen for an acknowledgment after a ping to transmit a packet containing sensed data with a timestamp.

Algorithm 2: Run-time EH Node

Input: C_{id}, $p\langle\varsigma, f_c\rangle$, τ_s, τ_a
Output: C_{id}, packets

wake up after τ_c or τ_s
ping C_{id} on f_c
while $E_r(t_s) > E_A$ *&&* $f_c \neq f_i$ **do**
  packet ← sense + timestamp
  listen for τ_a
  if *received ack.* **then**
    send (packet ⊕ SRAM)
    sleep for received τ_s
  end
  else
    SRAM ← packet
    sleep for τ_c
  end
end
switch to sleep core for τ_c

Notice that without a received acknowledgment, energy wastes are reduced. Specifically, when there are unheard packets, they are stored in local SRAM to be concatenated with packets made in further attempts. Hence, the EH node will not only save energy from avoiding data losses but also have prompt task scheduling to avoid idle listening while preserving data and creating purpose for transient on-board power with attempts to avoid full-capacity storage and not limiting harvested energy.

5 EXPERIMENTATION

In this section, conducted experimentation is shown to observe differences in throughput, amount of fault occurrence, and duration of healing after faults. Three trials are presented for a different number of nodes initialized to the network.

5.1 Experimental Settings

This subsection details the experimental setup of the selected hardware and software, as well as the methods taken for conducting the trials on experimentation for data acquisition.

5.1.1 Hardware. The selected components for the LoRa main gateway receiver as well as edge transmitters were several Heltec "LoRa Wi-Fi 32" radio modules. These embedded systems contain SX1276 LoRa radio chips as well as dual-core 32-bit "ESP32" CPU's for faster computation and enabling the sleep technology for providing options of very low current without powering out in sleep-states from the supporting ultra-low processing core. For voltage regulation as well as energy storage we use the "CJMCU-2557" energy harvester module for the efficient harvesting of energy from the integrated MPPT. Coupling the CJMCU-2557's output ports with the input pins of the radio module powers the radio and its "V_OK" port is used as an interrupt to trigger the initial wake-up. For the energy storage, we use (2.7V, 3.3F) SeekEC super-capacitors. The system used for acquiring these analog voltage measurements was a few voltage-current analog NI-9207 modules mounted securely on the compatible main NI-cDAQ chassis interfaced with the computer employed by the user for recording measurements. Several pins from the available 16 I/O analog pins on the NI-9207 modules were measured in parallel with the CJMCU-2557's input , super-capacitor, and output ports. A Sigilent SPD3303X-E and a Keithley 2231a-30-3 dc-power supply systems were used to provide enough independent channels for every node individually so that there wasn't current interference in nodes during run-time delivering insufficient approximate 2.2V voltage to model ambient sources.

5.1.2 Software. The Arduino IDE was used for programming and uploading the proposed algorithms to the utilized Heltec LoRa edge node modules to make use of the available LoRa and Heltec libraries available in the Arduino environment. The libraries used from open source available repositories include the HelTecAutomation [1], Espressif [4], and Sandeep Mistry [14] GitHub repositories. The functions used to implement the algorithms proposed using LoRa with the selected IDE such as booting, sensing, sleeping, parsing LoRa packets, locally setting frequency or spreading factor, etc. are provided from the supporting libraries mentioned. A project was created in the compatible NI-MAX software to have multi-channel analog readings and recording from the user data acquisition system and was used for plotting the measured values.

5.1.3 Procedure. The proposed algorithms are uploaded to the senders and gateway. The gateway establishes a local LoRa server and begins listening for the known N amount of nodes to allocate the unique resources to all the nodes in the network depending on the trial. The first set of trials has 3 nodes so the gateway is pre-programmed to know that the initialization won't be finished and all computations involving directed sleep times will be with respect to 3 nodes specifically so that the scheduled duty cycle begins immediately after the initialization finishes confidently. The second set of trials has the gateway pre-programmed with 4, and the third with with 5. After the gateway is preset to know how many nodes are expected to be initialized to the network, the user turns on the power supply and allows enough time for sufficient charging before performing hard resets on nodes for initialization. Then at random time periods post-initialization when the algorithms are running observe the performance of all nodes communicating during the networks duty cycle. This is done for 3-5 number of nodes for several trials each to obtain averages of throughput per duty cycle as well as amount of faults and time taken to heal.

5.2 Experimental Results

In Figures 3-5, three post-initialization runtime trials demonstrate uninterrupted scheduled cycles for various nodes within the duty.

Moderate operational voltage drops indicate scheduled wakes in duty cycles, with drops in higher magnitude representing interrupted wake-ups for healing requests. As shown, with the input voltages or supply delivering approximately 2.2V, insufficient to directly power the radios, the measured output voltages of the nodes as shown sustain all tasks and operations continuously within the safe operational voltage range of 3V-3.6V without degradation over time, even with an increased workload from additional nodes included in scheduled duty cycle. Furthermore, even in cases of faults as directed from the red arrows in the trials where nodes fall below safe operating range and undergo a power outage, after a brief time it is back in regular synchronized communication as directed

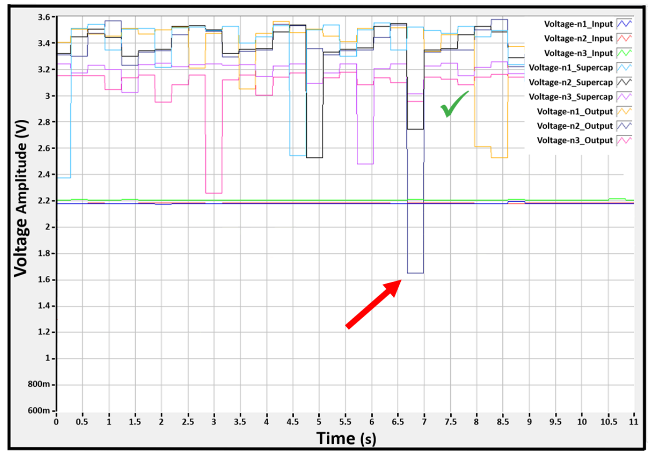

Figure 3: 3 node example trial

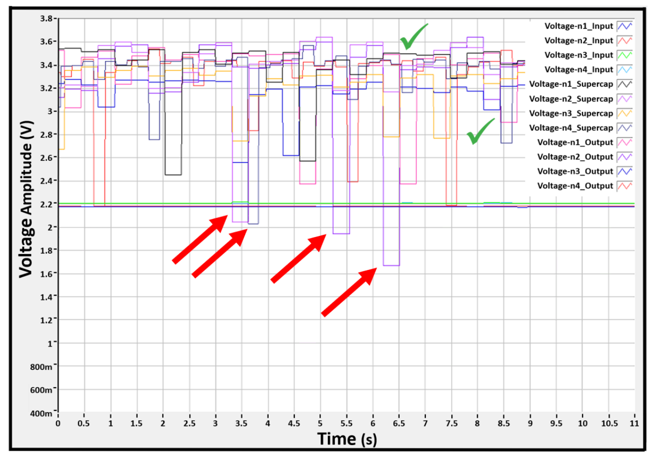

Figure 4: 4 node example trial

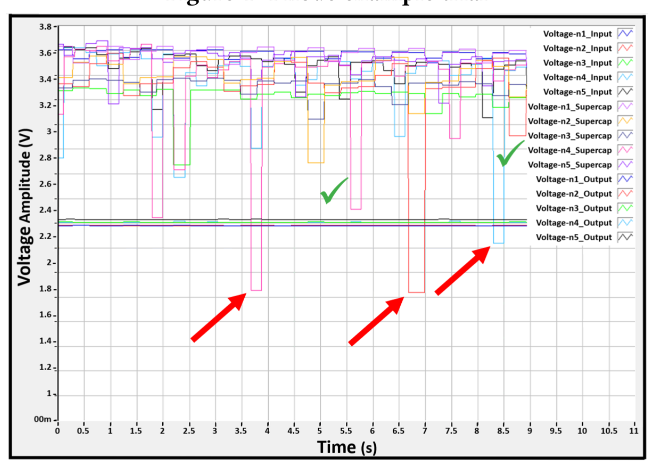

Figure 5: 5 node example trial

by the check-marks. This monitoring reveals that even if a node momentarily drops below the safe voltage range resulting to a fault, it swiftly returns to scheduled communication without interrupting other nodes' communication tasks.

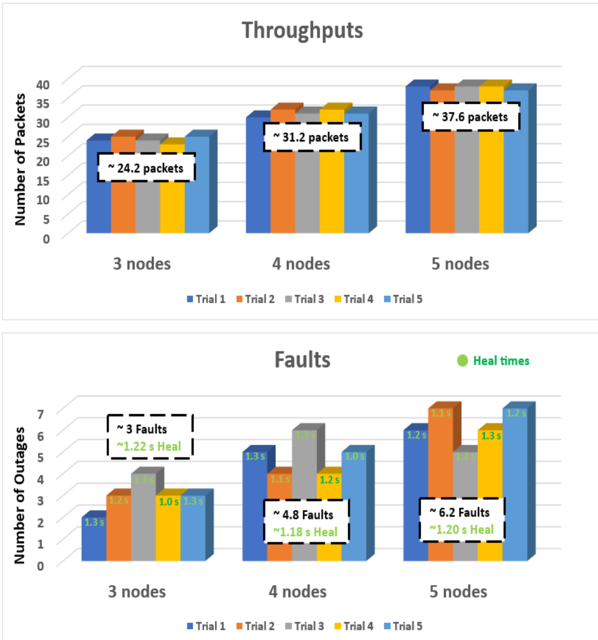

Figure 6: Average throughput, faults, and heal-time

5.3 Discussion

Average throughput of the system for the amount of trials is calculated by averaging the amount of received messages obtained serially from the gateway for every set of node amounts. The fault averages are calculated from averaging the amount of network requests the gateway received serially, and the average times taken to heal are from taking the differences of every fault to followed successfully scheduled communication. Figure 6 shows the obtained average throughput for every set of trials as well as the average amount of faults and average time duration for healing. Some trends or correlations that are realized with respect to average throughput per node and the average faults as well as average time taken for healing using the proposed algorithms are as follows:

- there was unfortunate increased fault probability with increased nodes, however the times taken to heal back to the network remains consistent regardless of the increased amount of nodes
- there were no interruptions with other nodes' communication in the scheduled duty cycle during the healing
- gateway was able to securely maximize throughput with additional nodes

The trials presented demonstrate the reliability of the proposed energy efficient task scheduling techniques in maintaining consistent fault tolerance with increased nodes and alternatively increasing throughput broadening the direction in further explorations for new techniques in increasing throughput in relative EH systems.

6 CONCLUSION

This paper introduces a self-sustaining IoT system comprising a gateway and EH edge nodes that receive unique resource allocations during initialization to prevent collisions. The system employs energy-efficient and waste-reduced task scheduling, locally preserving data until confidently acknowledged by the gateway, and maximizing throughput under limited power supply scenarios. We developed an energy-efficient task scheduling algorithm, combining power management techniques like anti-collision resource allocation, deadlock-breaking, MPPT-based energy harvesting, and fault tolerance. Additionally, it introduces a throughput-maximization strategy with temporary local storage to prevent data loss. Experimental results validate the method's robust reliability. Future work will involve evaluating the performance of these algorithms with complete energy harvesting systems supplying the power with ambient sensors for further observations as well as exploring methods for integrating dynamic UAV gateways.

REFERENCES

[1] HelTec Automation. 2024. Heltec_ESP32. https://github.com/HelTecAutomation/ESP32_LoRaWAN
[2] Nour El Hoda Djidi, Matthieu Gautier, Antoine Courtay, Olivier Berder, and Michele Magno. 2021. How Can Wake-up Radio Reduce LoRa Downlink Latency for Energy Harvesting Sensor Nodes? *Sensors* 21, 3 (2021). https://doi.org/10.3390/s21030733
[3] Ruben Dominguez, Wen Zhang, Hongzhi Xu, Pablo Rangel, and Chen Pan. 2023. ISSAC: An Self-organizing and Self-healing MAC Design for Intermittent Communication Systems. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*. 1–8. https://doi.org/10.1109/ISQED57927.2023.10129347
[4] espressif. 2024. esptool. https://github.com/espressif/esptool.git
[5] Golshan Famitafreshi, M. Shahwaiz Afaqui, and Joan Melià-Seguí. 2021. A Comprehensive Review on Energy Harvesting Integration in IoT Systems from MAC Layer Perspective: Challenges and Opportunities. *Sensors* 21, 9 (2021). https://doi.org/10.3390/s21093097
[6] Weiwei Fang, Xiaojie Zhao, Yuan An, Jing Li, Zhulin An, and Qiang Liu. 2016. Optimal scheduling for energy harvesting mobile sensing devices. *Computer Communications* 75 (2016), 62–70. https://doi.org/10.1016/j.comcom.2015.09.010
[7] Orestis Georgiou, Constantinos Psomas, Eleni Demarchou, and Ioannis Krikidis. 2021. LoRa Network Performance Under Ambient Energy Harvesting and Random Transmission Schemes. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. https://doi.org/10.1109/ICC42927.2021.9500756
[8] Rami Hamdi and Marwa Qaraqe. 2021. Resource Management in Energy Harvesting Powered LoRa Wireless Networks. In *ICC 2021 - IEEE International Conference on Communications*. 1–6. https://doi.org/10.1109/ICC42927.2021.9500638
[9] V. Joseph, Vinod Sharma, and U. Mukherji. 2009. Optimal Sleep-Wake Policies for an Energy Harvesting Sensor Node. *ICC'09*, 1 – 6. https://doi.org/10.1109/ICC.2009.5199286
[10] Meng-Lin Ku, Wei Li, Yan Chen, and K. J. Ray Liu. 2016. Advances in Energy Harvesting Communications: Past, Present, and Future Challenges. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1384–1412. https://doi.org/10.1109/COMST.2015.2497324
[11] Mahmuda Khatun Mishu, Md. Rokonuzzaman, Jagadeesh Pasupuleti, Mohammad Shakeri, Kazi Sajedur Rahman, Fazrena Azlee Hamid, Sieh Kiong Tiong, and Nowshad Amin. 2020. Prospective Efficient Ambient Energy Harvesting Sources for IoT-Equipped Sensor Applications. *Electronics* 9, 9 (2020). https://doi.org/10.3390/electronics9091345
[12] Chen Pan, Mimi Xie, Yongpan Liu, Yanzhi Wang, Chun Jason Xue, Yuangang Wang, Yiran Chen, and Jingtong Hu. 2017. A Lightweight Progress Maximization Scheduler for Non-Volatile Processor under Unstable Energy Harvesting. *SIGPLAN Not.* 52, 5 (jun 2017), 101–110. https://doi.org/10.1145/3140582.3081038
[13] Merin Susan Philip and Poonam Singh. 2020. Review of Energy Harvesting in LoRa based Wireless Sensor Network. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. 338–341. https://doi.org/10.1109/ICACCCN51052.2020.9362892
[14] sandeepmistry. 2024. arduino-LoRa. https://github.com/sandeepmistry/
[15] Semtech Corporation 2020. *SX/1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver.* Semtech Corporation. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE
[16] Tong Wang, Xiang Yang, Feng Deng, Lin Gao, Yufei Jiang, and Zhihua Yang. 2020. Joint power and duty-cycle design using alternating optimization algorithm under energy harvesting architectures. *China Communications* 17, 12 (2020), 139–155. https://doi.org/10.23919/JCC.2020.12.010
[17] Rongrong Zhang, Jihong Yu, Yong Guan, and Jiangchuan Liu. 2021. A Dominating Set-Based Sleep Scheduling in Energy Harvesting WBANs. *IEEE Transactions on Vehicular Technology* 70, 11 (2021), 11923–11934. https://doi.org/10.1109/TVT.2021.3085833