Green Technologies and Sustainability 4 (2026) 100409

Contents lists available at ScienceDirect

Green Technologies and Sustainability

journal homepage:
https://www.keaipublishing.com/en/journals/green-technologies-and-sustainability/

Full-length article

Optimizing communication and computational cost for IoT devices in energy efficient swarm robotics

Amir Ijaz *, Hashem Haghbayan, Ethiopia Nigussie, Abdul Malik, Juha Plosila

Department of Computing, University of Turku, Finland

HIGHLIGHTS

- Validate energy-efficient IoT operation through detailed simulations.
- Analyze communication energy factors in robotic IoT swarms.
- Relate computation and communication costs in swarm scheduling.
- Propose a heuristic for joint workload and communication optimization.
- Implement adaptive wireless protocols on embedded hardware.

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Protocols
Edge devices
Resource management
Power consumption
Optimization

ABSTRACT

The Internet of Things (IoT) renders swarm robotics possible, which makes tasks like surveillance, farming, and disaster response more efficient and flexible. The limited energy and processing power on board make things very difficult, especially since coordination requires a lot of communication. The goal of this paper is to lower the cost of communication in robotic swarms so that they last longer and make better use of resources. We offer a unified optimization framework that combines adaptive communication protocols with heuristic offloading algorithms. The main goal is to reduce communication energy, with computational cost being a secondary concern. The system changes the power levels and transmission rates based on how well the network is working. It uses a heuristic based on PSO to find the best balance between processing data locally and sending it to the cloud. Numerous evaluations on embedded platforms (NVIDIA Jetson Nano and TX2) demonstrate that the proposed method significantly conserves energy, reducing communication related energy consumption by approximately 25–35 percent relative to static schemes, while also mitigating CPU load fluctuations. Our results show that putting communication optimization first greatly improves swarm energy efficiency without hurting coordination performance.

1. Introduction

Adding IoT devices to swarm robotics lets many robots work together on difficult tasks, which is useful for everything from monitoring the environment to automating factories [1]. In such systems, each robot has limited battery life and onboard processing capability [2]. Notably, *communication among swarm robots* can consume a large fraction of total energy [3]. In fact, studies report that up to 80 percent of energy in IoT sensor networks is spent on wireless data transmission [4]. Thus, minimizing communication overhead is critical to extending the swarm's operational time [5].

Prior work has explored various strategies for energy efficient swarm coordination [6]. Cluster based communication topologies, for example, have been shown to substantially lower execution time and battery usage compared to full mesh networking [7]. Similarly, adaptive transmission schemes using learning algorithms can dynamically balance energy versus reliability [8]. However, many approaches treat communication and computation separately [9]. We propose a holistic framework that explicitly prioritizes communication energy optimization, while still accounting for local processing cost (See Fig. 1).

The scope of this work is to develop and evaluate a communication aware optimization framework for IoT enabled swarm robotics, where communication energy is treated as the primary bottleneck and computational cost is jointly considered under battery, CPU, and bandwidth constraints. The paper makes three main contributions: (i) a unified model that captures the coupled communication-computation trade off in heterogeneous swarms, (ii) a practical optimization strategy that

* Corresponding author.
E-mail address: amir.ijaz@utu.fi (A. Ijaz).

https://doi.org/10.1016/j.grets.2026.100409
Received 3 March 2026; Received in revised form 4 May 2026; Accepted 5 May 2026

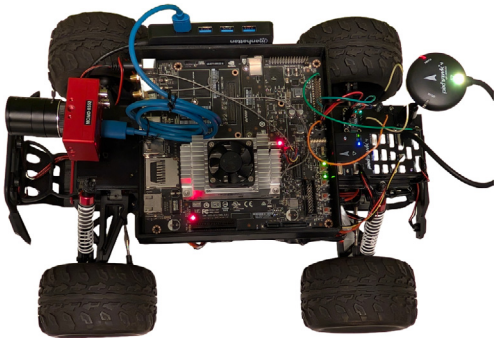

Fig. 1. Prototype of robotics platform used in our experiments.

prioritizes communication energy reduction while enforcing feasibility, and (iii) an experimental validation on embedded platforms showing measurable reductions in communication energy and computational load. Future work is restricted to extensions of this same framework, including more accurate energy models, stronger constraint handling, and deployment under larger and more dynamic swarm conditions.

One of the biggest problems is finding ways to save energy for groups of robots that work in different environments [10]. For robotic systems to work in the future, they need to be able to operate in swarms that use less energy [11]. The biggest problem is figuring out how to schedule tasks and form robots while using as little energy as possible [12]. The battery drains energy from the robot's sensors, electronic systems, and mechanical engines, among other parts [13]. The amount of energy each part uses varies a lot depending on things like how many sensors it has, how powerful its computer is, how it communicates, and how big it is [14]. So, to get the most out of all the energy used, we need to look at all the parts of the system.

Research has previously concentrated on the optimization of individual robot components in isolation, including mechanical energy optimization, computational energy efficiency, sensor gating, and low power communication [15]. However, in robotics, the interdependencies among these components mean that decisions made for one part can influence the operations of others [16]. For instance, a decision on the robot's path, i.e., a mechanical choice, can affect the required perception data and the computational load for processing it [17]. Moreover, limited computational capabilities may compromise accuracy and decision making quality [18].

These interdependencies underscore the importance of co-optimizing interconnected components [19]. Some studies have proposed co-optimization strategies for the computational and mechanical components of robots [20]. In swarm robotics, these interdependencies become even more pronounced due to task sharing among nodes and the costs associated with communication [21]. Consequently, energy efficient resource management in swarms is significantly more complex than in single robots [22].

This paper proposes a co-optimization framework for the cyber and mechanical components within a swarm. We demonstrate that independent optimization of cyber components (computation and communication) and mechanical components does not necessarily yield the lowest total energy expenditure. Instead, we illustrate the necessity of aligning mechanical decision making with computational workload distribution and communication costs among nodes, leading to substantial energy savings for the swarm. The main contributions of this paper include:

- **A joint communication-computation optimization model for heterogeneous swarm robots.** We formulate a swarm level resource allocation problem that explicitly couples communication energy and local computation energy through the terms $E_i^{\text{comm}} = \xi_i \theta_i B_i$ and $E_i^{\text{comp}} = \eta_i(1-\theta_i)L_i$, while enforcing per node battery, CPU, and shared bandwidth constraints. This makes the communication-computation trade off explicit rather than treating communication as an auxiliary cost.
- **A communication prioritized objective for swarm offloading.** We introduce a weighted objective

$$J = \sum_{i=1}^{N} \left(w_c E_i^{\text{comm}} + w_p E_i^{\text{comp}} \right), \qquad w_c > w_p,$$

  so that the optimizer explicitly prioritizes reducing wireless transmission cost when bandwidth and energy are limited. This choice reflects the observation that communication often dominates the energy budget in swarm deployments.
- **A PSO based feasible offloading algorithm with constraint handling.** We develop an iterative PSO heuristic that updates offloading fractions θ_i using local and global best solutions, while enforcing feasibility through projection onto $[0, 1]$, bandwidth scaling, and per robot CPU and energy budget corrections. This yields a practical near optimal solver for large heterogeneous swarms where exact optimization would be expensive.
- **An adaptive communication layer integrated with the optimization loop.** We combine the offloading strategy with adaptive wireless control that adjusts transmission power, data rate, and duty cycle according to residual energy and network conditions. This reduces redundant messaging and limits contention on shared links, directly lowering communication energy.
- **Experimental validation on heterogeneous embedded platforms.** We validate the framework on Jetson Nano and Jetson TX2 platforms using representative swarm workloads and communication settings. The experiments show approximately 25–35% reduction in communication energy, lower average and peak CPU utilization, and improved task completion behavior under the optimized policy.

Overall, the paper contributes a communication aware resource management framework that is both algorithmically practical and experimentally grounded, showing that prioritizing communication energy yields measurable gains in swarm lifetime and operational efficiency.

The rest of the paper is structured as follows. Section 2 reviews related work. Section 3 describes the system model and formulates the optimization problem. Section 4 gives the details of the implementation. Section 5 shows the results of the evaluation and a discussion of them. Finally, Section 6 wraps up the paper and talks about what could come next.

2. Related work

Different algorithms and protocols have been used to make swarm robotics more energy efficient [23]. Particle swarm optimization (PSO) inspired distributed task allocation methods have been used to balance workloads and cut down on idle time, which saves energy in an indirect way [24]. Communication optimizations, such as clustered messaging, have been specifically shown to reduce both latency and power consumption by limiting redundant broadcasts [25]. Reinforcement learning approaches have been applied to adapt transmission power and duty cycles in dynamic IoT networks, achieving up to 25 percent longer device lifetimes [26]. Data aggregation and compression techniques also help reduce transmitted payloads, which is especially important given that up to 80 percent of IoT node energy can go to communication [27]. Unlike most prior work that considers power management and communication separately, our framework jointly optimizes communication scheduling and computation offloading, with explicit emphasis on minimizing communication energy.

Research relevant to joint communication and computation optimization spans multiple literatures: energy aware wireless protocols and data reduction, computation offloading and task allocation

(edge/fog), heuristic and learning based resource management, and energy aware swarm robotics. Below we summarize representative contributions, highlight methodological choices, and identify gaps addressed by the present work.

A large body of work targets energy reduction at the link and protocol layers through adaptive power control, duty cycling, MAC scheduling, and protocol selection [28]. Surveys and comparative studies in the IoT/edge domain describe how protocol parameters (e.g., transmit power, modulation, duty cycle) and application layer strategies (aggregation, compression) reduce transmission cost in practice [29]. LPWAN technologies (LoRaWAN, NB-IoT) and short range protocols (BLE) have been benchmarked for energy per bit, latency and reliability trade offs; these studies show that per bit energy is strongly influenced by protocol overhead and retransmission behavior, motivating payload aware transmission policies. However, many communication focused works treat computation cost as exogenous, neglecting the trade off that arises when computational offloading can reduce message volume or change latency/energy trade offs [30].

Computation offloading to edge or cloud resources is a well studied approach to reduce device side computation energy and accelerate processing. Classic formulations model offloading as optimization problems balancing local processing energy/time vs. transmission energy/time under latency and energy constraints. Some works propose analytic convex formulations that yield globally optimal offloading fractions for divisible workloads, while others consider discrete task allocation with combinatorial complexity and adopt heuristics or mixed integer programming [31]. Offloading research often assumes a stable, high capacity edge and does not explicitly model the impact of multiple devices contending for shared, constrained wireless resources typical in swarm scenarios.

A more recent trend explicitly couples communication and computation costs. Joint optimization frameworks incorporate both per bit transmission cost and per cycle computation cost in a unified objective, subject to CPU, energy and network constraints [32]. These formulations demonstrate that substantial gains arise from co-design: e.g., selectively compressing or batching data before transmission, or offloading selectively based on network state. Nevertheless, exact solutions scale poorly with swarm size and heterogeneity, motivating scalable heuristics and approximations.

Heuristic meta-heuristics (PSO, genetic algorithms, simulated annealing) and machine learning approaches (supervised, RL) have been proposed to solve large scale or dynamic instances. PSO has been applied to resource allocation and task scheduling because of its simplicity and parallelisability. Reinforcement learning (RL), including Deep Q-Networks and actor critic methods, has been used for adaptive power control and duty cycle management in nonstationary channels. These methods adapt to runtime conditions but face sample efficiency, safety (exploration) and deployment cost challenges on constrained hardware; hybrid designs that combine light weight heuristics with learning have been recommended.

All of the robots in swarm robotics are different, they need to be coordinated by communicating a lot, and missions can only last so long before the robots run out of power. Previous research on swarm systems has looked at how to assign tasks while taking energy into account, how to control formations when energy is limited, and how to use duty cycling to make swarms last longer. A lot of swarm studies, on the other hand, focus on control and coordination algorithms that do not have detailed energy models or on heuristics that make communication less effective but do not take into account how to systematically optimize both communication and computational costs [33]. There are not yet a lot of real world tests on embedded platforms like the Jetson Nano and TX2 that show how the trade offs between communication and computation work in the real world.

The integration of IoT technologies into swarm robotics has led to extensive research on energy aware communication, computation offloading, and resource allocation. Prior work has shown that adaptive power control, duty cycling, and protocol level optimization can reduce wireless energy consumption in constrained devices, while protocol selection and data reduction strategies can further improve energy efficiency in IoT edge systems [34]. At the same time, computation offloading has been widely studied as a means to balance local processing energy against transmission cost under CPU, battery, and latency constraints. More recent studies have considered joint optimization of communication and computation, demonstrating that co-design can yield significant energy savings, although such formulations often become difficult to solve at swarm scale. Heuristic and learning based methods, including PSO and reinforcement learning, have therefore been proposed to improve scalability and adaptivity in dynamic environments. In swarm robotics specifically, the literature has investigated energy aware task allocation, formation control, and duty cycling, but comparatively fewer works explicitly co-optimize communication and computation under realistic embedded hardware constraints. These gaps motivate the present study, which focuses on communication aware resource optimization for heterogeneous swarm robotic platforms.

The research literature shows big steps forward in many areas, including communication, computation, heuristics, learning, and swarm level design. However, there are still big issues to address:

- **Joint modeling on a scale that is true to life:** The number of studies that show that joint comm-comp optimization works with real workloads is low. This research examines various types of embedded hardware and unreliable wireless connections.
- **Scalable, adaptive algorithms:** Exact optimization does not work for big swarms, so we need algorithms that can grow, and enforce strict limits (like bandwidth, CPU, and battery).
- **Safety aware learning:** RL methods need to work with safe backups for energy critical systems. Fully autonomous learning without protections is dangerous in deployed swarms.
- **Protocol aware offloading:** The interplay between application layer protocols (MQTT or CoAP), LPWANs, and offloading decisions is underexplored.

Our work addresses these gaps by (i) proposing an integrated optimization framework that explicitly weights communication and computation energy, (ii) combining scalable heuristics (PSO) with adaptive communication and safe, on device learning, and (iii) validating the framework on heterogeneous embedded platforms and representative protocols. This places our approach at the intersection of practical protocol aware engineering and principled optimization for energy efficient swarm robotics.

3. System model and optimization framework

We model a swarm of N IoT enabled robots. Each robot i has a finite battery and performs sensing/computation tasks. Let B_i be the data size (in bits) of tasks for robot i, and L_i the number of CPU cycles required to process this data locally. We denote by $\theta_i \in [0, 1]$ the fraction of data that robot i offloads to an edge server or another robot, and $(1 - \theta_i)$ the fraction processed locally.

Fig. 2 illustrates the proposed system model and optimization framework for energy efficient swarm robotics. The architecture is composed of a swarm of heterogeneous, energy constrained robots (e.g., Jetson Nano and Jetson TX2 platforms) that perform local sensing, computation, and wireless communication while interacting with an adaptive, communication aware optimization layer.

At the swarm layer, each robot is characterized by its task data size B_i, computational workload L_i, and finite energy budget $E_i^{\max}$. These parameters set the local operating limits that robots must use to decide whether to do tasks locally or send them to another location via wireless links. The diagram elucidates the close relationship between communication and computation. More offloading lowers the costs of

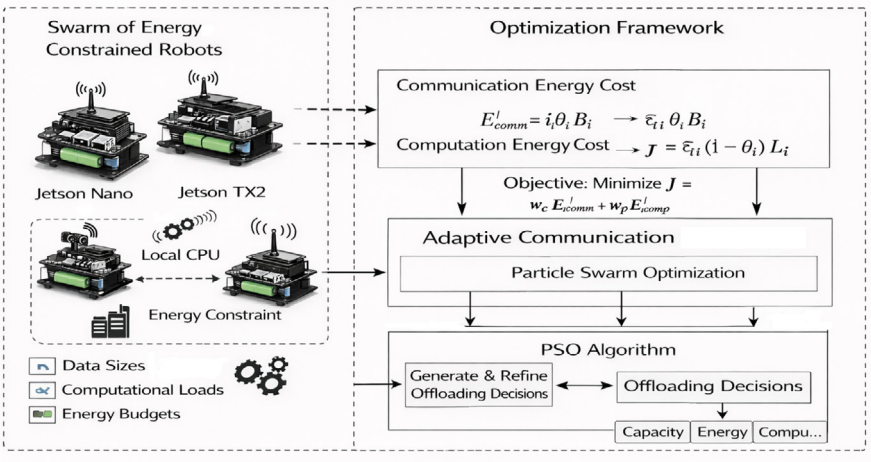

Fig. 2. System model and proposed optimization framework.

processing data locally, but it raises the costs of sending data, which is often the biggest part of the total energy use in distributed robotic systems.

Communication and computation use different amounts of energy, which are shown clearly by the optimization layer: $E_i^{\text{comm}} = \xi_i \theta_i B_i$ and $E_i^{\text{comp}} = \eta_i(1-\theta_i)L_i$. These terms are included in a global objective function that has weights.

$$J = \sum_{i=1}^{N} \left(w_c E_i^{\text{comm}} + w_p E_i^{\text{comp}} \right) \tag{1}$$

where $w_c > w_p$ tilts the optimization toward using less energy for communication. This formulation shows that in big swarms, the costs of wireless transmission are often higher than the costs of local computation. This makes communication aware scheduling an important way to keep systems running for a long time.

The adaptive communication module modifies transmission parameters such as data rate and duty cycle in accordance with the remaining energy and the network load. The framework limits the growth of aggregate communication energy, especially when there are a lot of robots deployed, by cutting down on unnecessary broadcasts and messaging between robots.

We use a Particle Swarm Optimization (PSO) heuristic to solve the constrained offloading problem in a way that is both distributed and scalable. Each robot improves its offloading decision θ_i over time by using feedback from both local and global performance, while also following system level rules about channel capacity, computational limits, and energy budgets for each node. The two-way flow between the "Generate and Refine Offloading Decisions" and "Offloading Decisions" blocks in the schematic shows how this process of constant feedback works.

In general, the framework creates a closed loop between the optimization layer and the physical swarm dynamics. The proposed model enables heterogeneous robots to cooperate efficiently by jointly adapting communication behavior and computational offloading under explicit energy and capacity constraints. This prioritizes reducing communication costs, which improves scalability and operational endurance in energy limited swarm deployments.

Fig. 3 shows the overall structure of the suggested framework for reducing both communication and computation costs in swarm robotics that have limited energy. The model breaks down system intelligence into two tightly linked control layers: an adaptive communication layer and a PSO based computational offloading layer. Both layers are controlled by a single energy centric objective function.

The adaptive communication module automatically changes the transmission power and data rates based on the current network load

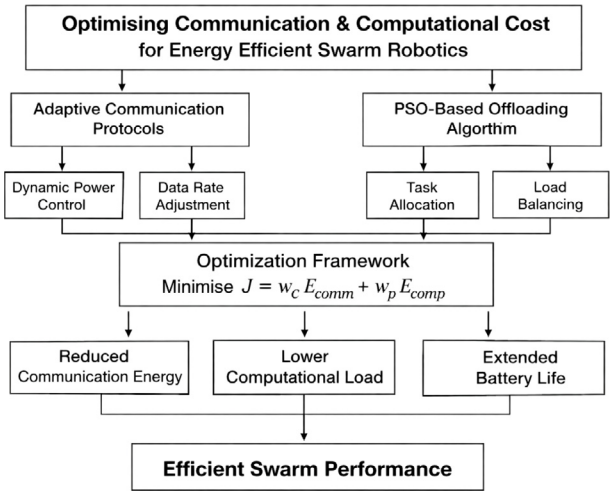

Fig. 3. Schematic for proposed optimization.

and the amount of battery left. The framework stops unnecessary messaging and limits contention on shared wireless channels by explicitly controlling the volume and quality of data exchange between robots. This system is engineered to manage the energy of communication. In dense or highly coordinated swarms, communication energy typically constitutes the predominant portion of the overall energy footprint.

The PSO based offloading algorithm ensures that each category of robot is assigned an equitable workload. Each agent frequently alters its decision regarding offloading, influenced by the swarm level and the local level's performance indicators. This enables the system to reach a state of reduced energy consumption. It elucidates the methodology for distributing tasks to achieve two objectives: reducing the workload on local computers and preventing the emergence of communication bottlenecks that would increase data transmission costs.

Within the primary optimization framework, two control pathways operate in conjunction. The weighted energy function is minimized to achieve this objective:

$$J = w_c E_{\text{comm}} + w_p E_{\text{comp}} \tag{2}$$

where w_c is selected to prioritize communication with reduced power when battery life and bandwidth are constrained. This facilitates the system's evaluation of the advantages and disadvantages of reduced computation relative to the expenses associated with increased transmission. Offloading will occur solely when it results in overall energy conservation.

The diagram indicates that assigning tasks that consider communication enhances productivity threefold. It requires reduced power for nodes to communicate, thereby extending the longevity of processors and batteries. When the swarm is established for an extended duration or across an extensive area, these modifications enhance its stability and scalability. This is particularly applicable when rapidity and energy efficiency are paramount.

A closed loop control system that does not use much energy is set up by the architecture. Collaboration and offloading of computation work together to make things work better. The proposed framework is different from other load balancing methods because it directly takes communication costs into account when it comes to optimization. This means that swarms can work well and reliably even when there are strict limits on energy and network.

The novelty of this work is *not* the use of Particle Swarm Optimization (PSO) itself; rather, PSO is used as a lightweight and scalable solver for a problem that is explicitly reformulated around *communication aware* resource allocation in energy constrained swarm robotics. As noted in the literature, exact joint comm-comp optimization becomes impractical as swarm size and heterogeneity increase, which motivates heuristic approximations such as PSO for large scale or dynamic instances. The main contribution of this paper is therefore the *problem formulation and system level integration*, not the optimizer alone.

More specifically, the proposed framework introduces the following novel points:

- **Communication first objective design.** The optimization explicitly weights communication energy more heavily than computation energy, i.e.,

$$J = \sum_{i=1}^{N} \left(w_c E_i^{\text{comm}} + w_p E_i^{\text{comp}} \right), \qquad w_c > w_p,$$

  so that the solution prioritizes reducing wireless transmission cost, which is often the dominant energy component in IoT enabled swarm operation.
- **Joint modeling of communication, computation, and resource constraints.** Unlike approaches that optimize only power or only scheduling, our model couples communication energy, local computation energy, per node battery limits, CPU capacity limits, and shared network capacity within a single optimization framework. This makes the communication-computation trade off explicit rather than implicit.
- **Adaptive communication control integrated with offloading.** The framework does not merely select offloading fractions; it also adapts transmission power, data rate, and duty cycle according to residual energy and network state. This is important because our results show that communication related energy can be reduced by approximately 25–35% when adaptive communication and offloading are combined.
- **Validation on heterogeneous embedded hardware and protocol relevant workloads.** The method is evaluated on Jetson Nano and Jetson TX2 platforms with representative swarm workloads and communication settings. The experiments show reduced CPU utilization, smoother load distribution, and improved task completion behavior alongside lower communication energy.
- **Practical swarm level relevance.** Prior swarm robotics studies often focus on task allocation or control in isolation, or on communication reduction without systematically co-optimizing computation and network usage. Our framework bridges that gap by coupling mechanical/task level decisions with cyber resource allocation in one closed loop design.

In summary, the contribution of this paper is the *communication aware co-optimization framework* for swarm robotics, together with its experimental validation. PSO is the enabling heuristic, but the scientific novelty lies in the joint formulation, the communication prioritized objective, the adaptive protocol layer, and the demonstrated system level gains in energy efficiency and operational endurance.

3.1. Energy consumption model

We use simple linear energy models for communication and computation. The energy to transmit B_i bits from robot i is modeled as

$$E_i^{\text{comm}} = \xi_i B_i \tag{3}$$

where ξ_i (Joules/bit) captures transmission power and channel conditions. If robot i processes data locally, its CPU energy is

$$E_i^{\text{comp}} = \eta_i L_i \tag{4}$$

where η_i (Joules/cycle) is the processor's energy coefficient. For offloading, the energy depends on θ_i: the robot sends $\theta_i B_i$ bits (consuming $\xi_i \theta_i B_i$ energy) and computes $(1-\theta_i)L_i$ cycles (consuming $\eta_i(1-\theta_i)L_i$). Thus,

$$E_i^{\text{comm}}(\theta_i) = \xi_i, \theta_i B_i, \;\; E_i^{\text{comp}}(\theta_i) = \eta_i, (1-\theta_i)L_i \tag{5}$$

We emphasize that optimizing energy consumption in isolation may lead to degenerate solutions (e.g., concentrating all computation on a single node) if no performance constraints are enforced. Therefore, we explicitly incorporate *execution time and quality of service (QoS) constraints* into the optimization framework. In particular, task latency, deadline satisfaction, and load balancing are introduced as additional objectives and constraints, ensuring that the solution remains practically relevant for distributed swarm operation. The formulation prevents trivial solutions by enforcing parallelism requirements and bounded task completion time.

While energy efficiency is the primary focus of this work, the optimization problem is formulated to jointly consider performance constraints to avoid degenerate solutions. Specifically, we augment the objective function with a latency aware term and enforce task execution constraints across the swarm. The global objective is therefore expressed as

$$J = \sum_{i=1}^{N} \left(w_c E_i^{\text{comm}} + w_p E_i^{\text{comp}} \right) + w_t T_{\text{tot}},$$

where T_{tot} denotes the overall task completion time (or makespan), and w_t is a weighting factor that ensures that latency is not neglected.

In addition, we impose the following constraints:

$$T_i \le T_i^{\max}, \quad \forall i \in \{1, \dots, N\}, \tag{6}$$

$$\sum_{i=1}^{N} \mathbb{1}\{(1-\theta_i)L_i > 0\} \ge N_{\min}, \tag{7}$$

where T_i is the execution time of task i, $T_i^{\max}$ is its deadline, and the second constraint enforces a minimum level of distributed participation across the swarm. This prevents the optimizer from collapsing all computation onto a single node and ensures that the swarm operates in a distributed manner consistent with its intended application.

Furthermore, the considered application scenarios (e.g., distributed perception and cooperative sensing) inherently require spatially distributed processing and timely response, which makes centralized execution infeasible due to communication latency, bandwidth limitations, and robustness considerations. By incorporating execution time into both the objective and constraints, the proposed framework ensures a balanced trade off between energy efficiency and system level performance, thereby aligning the optimization with realistic swarm robotics requirements.

3.2. Constraints

Robots are subject to resource limits. Each has a battery energy budget $E_i^{\max}$, yielding

$$E_i^{\text{comm}}(\theta_i) + E_i^{\text{comp}}(\theta_i); \le; E_i^{\max}, \quad \forall i \tag{8}$$

There is also a maximum local CPU capacity $C_i^{\max}$ (cycles per time slot) so that $(1-\theta_i)L_i \le C_i^{\max}$ for each i. Finally, the communication channel

has limited aggregate capacity. If the shared uplink rate allows $B^{\max}$ bits per cycle, we enforce

$$\sum_{i=1}^{N} \theta_i, B_i; \leq; B^{\max} \tag{9}$$

so that total offloaded data does not exceed the channel capability.

We clarify that our framework may be extended with a *look ahead* component that estimates near future computation and communication demand over the next control window. This prediction can be used together with the current residual energy state to improve offloading and communication decisions, particularly for workloads with structured execution such as vision pipelines or multi stage cooperative tasks.

Our proposed framework is not limited to past energy observations. In addition to the current residual energy $E_{\text{res}}(t)$, we incorporate a short horizon prediction of the upcoming workload, denoted by $\widehat{L}(t+1)$, and, where relevant, the expected communication demand $\widehat{B}(t+1)$. These predictions can be obtained from the structure of the running algorithm, task graph, or recent execution history. For example, in a vision pipeline, the current processing stage can indicate whether the next interval will require feature extraction, matching, or classification, each of which has different computational and communication requirements.

Accordingly, the offloading decision at time t is based on both the current and anticipated resource requirements:

$$\theta_i(t) = f\Big(E_{\text{res}}(t),\ \widehat{L}(t+1),\ \widehat{B}(t+1),\ q(t),\ \text{SNR}(t)\Big),$$

where $q(t)$ denotes queue state or pending workload. This makes the optimization predictive rather than purely reactive, allowing the system to reserve local or network resources for imminent demand. If the next control interval is expected to be computation intensive, the framework can avoid unnecessary offloading that would create congestion or delay. Conversely, if future demand is low, the controller can enter a more conservative communication mode and preserve energy.

To reflect this in the objective, the optimization uses a forecast aware cost:

$$J(t) = w_c E_i^{\text{comm}}(t) + w_p E_i^{\text{comp}}(t) + w_f \widehat{E}_i^{\text{future}}(t+1),$$

where $\widehat{E}_i^{\text{future}}(t+1)$ represents the predicted energy impact of the next execution window. This look ahead term improves decision quality when workloads are sequential or partially structured, and it reduces the risk of myopic actions that are energy optimal in the current step but inefficient over the full task horizon.

In summary, the framework uses both present state measurements and short horizon workload prediction to guide offloading and communication control. This predictive extension better matches the dynamics of real swarm algorithms, where current execution state provides information about near future computation and communication needs.

3.3. Optimization problem

Our goal is to find offloading fractions θ_i that minimize energy. We define the weighted sum of total communication and computation energy:

$$J; =; \sum_{i=1}^{N} \big(w_c, E_i^{\text{comm}}(\theta_i) + w_p, E_i^{\text{comp}}(\theta_i)\big) \tag{10}$$

where weights $w_c, w_p \geq 0$ reflect the importance of communication vs computation energy. To prioritize communication cost, we choose $w_c > w_p$. The optimization is:

$$\text{J} = \sum_{i=1}^{N} \big(w_c, \xi_i, \theta_i B_i; +; w_p, \eta_i(1-\theta_i)L_i\big) \text{ subject to } \quad 0 \leq \theta_i \leq 1 \tag{11}$$

This linear program yields an optimal split θ_i^* for each robot. Typically, a robot will offload tasks until the marginal communication cost equals the marginal computation cost, subject to constraints. Our framework allows emphasis on communication by setting $w_c > w_p$. For example, in a scenario with scarce bandwidth or high transmission cost, choosing a larger w_c discourages offloading and thus reduces communication energy.

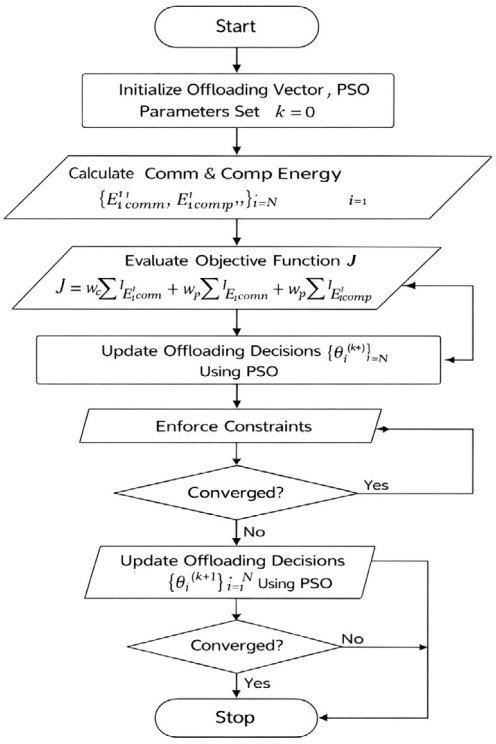

Fig. 4. Proposed methodology.

4. Implementation

We implemented the framework on hardware representative of swarm robots and integrated communication protocols and heuristics in software.

Fig. 4 details the iterative optimization workflow employed to jointly minimize communication and computational energy in the proposed swarm robotics framework. The procedure begins with the initialization of the offloading vector $\theta^{(0)}$ and the Particle Swarm Optimization (PSO) control parameters, establishing the initial distribution of local execution versus task offloading across the swarm.

At each iteration k, the framework evaluates the per robot communication and computation energy components, $\{E_{i,\text{comm}}^{(k)}, E_{i,\text{comp}}^{(k)}\}_{i=1}^{N}$, based on current task sizes, channel conditions, and processing loads. These energy terms are aggregated into the global objective function

$$J^{(k)} = w_c \sum_{i=1}^{N} E_{i,\text{comm}}^{(k)} + w_p \sum_{i=1}^{N} E_{i,\text{comp}}^{(k)} \tag{12}$$

where w_c and w_p control the relative emphasis on communication and computational efficiency, respectively. In keeping with the framework's communication aware design, w_c is chosen to be the most important factor when bandwidth is limited or energy is critical.

The PSO update stage improves the offloading choices $\theta^{(k)}$ by putting together the best local and global solutions. This helps the system find ways to assign tasks that use less energy. This method of searching lets each robot change how it acts based on what the whole group says, without having to talk to or share information with the whole group. In this way, the devices can talk to each other for a lower cost.

There are limits on how much power each node can use, how many wireless channels can be used, and how much computing power is available. Forced constraints make sure that every new idea fits within these limits. Adding feasibility checks to the iterative loop stops the framework from getting to operating points that are energetically best but not possible in real life.

The convergence test creates a closed loop control system that keeps improving until the global goal is only slightly better after each update or until a certain number of iterations have been completed. This termination criterion weighs the quality of the solution against the extra work that needs to be done on computing and communication. This makes sure that the optimization process does not undo the energy savings it is trying to make.

We define the offloading destination as an *available peer node, cluster head, or edge server* with sufficient residual energy, CPU capacity, and link quality. If no feasible target exists, the robot retains the task locally and the framework sets the offloading fraction to zero for that decision epoch. We also now account for the computation required by the offloading decision itself as a lightweight control overhead term in the optimization model, so that the control cost is not ignored.

In the proposed framework, computation is offloaded from a robot either to a nearby peer robot, a designated leader/cluster head node, or an edge server, depending on current resource availability and communication quality. Let

$$\mathcal{D}_i(t) = \left\{ j \in \mathcal{N}_i(t) \;:\; E_j^{\mathrm{res}}(t) \geq E_{\min},\; C_j^{\max}(t) \geq L_i(t),\; \mathrm{SNR}_{ij}(t) \geq \mathrm{SNR}_{\min} \right\}$$

denote the set of feasible offloading targets for robot i at time t, where $\mathcal{N}_i(t)$ includes neighboring robots and, when available, an edge server. The offloading decision is then made only if $\mathcal{D}_i(t) \neq \emptyset$. If no feasible receiver is available, the framework falls back to local execution, i.e., $\theta_i(t) = 0$, thereby guaranteeing that the task remains schedulable even in congested or energy critical conditions.

For the considered vision workloads, such as image classification, corner detection, and image filtering, the task may be forwarded to the nearest feasible peer or to the edge server if the peer set is empty or insufficiently capable. This makes the offloading policy explicit and avoids ambiguity in the execution path. The objective therefore captures not only the energy required for communication and local computation, but also the feasibility of the target node that receives the offloaded task.

To account for the overhead of the decision process itself, we include a small control cost term $E_i^{\mathrm{dec}}(t)$, representing the energy required for state observation, feasibility checking, and offloading selection. The total energy associated with a decision epoch is therefore written as

$$E_i^{\mathrm{tot}}(t) = E_i^{\mathrm{comm}}(t) + E_i^{\mathrm{comp}}(t) + E_i^{\mathrm{dec}}(t).$$

Since the optimization is executed at a slower control rate than the data transmission process, $E_i^{\mathrm{dec}}(t)$ is lightweight relative to the communication term and does not alter the overall energy trend; however, including it makes the optimization model complete and avoids neglecting control overhead. If no feasible offloading target exists, the optimization naturally resolves to local execution, ensuring robustness under limited connectivity and preserving task continuity.

Our proposed algorithm 1 solves a joint communication computation offloading problem for a swarm of N robots by selecting per robot offloading fractions $\theta_i \in [0, 1]$ to minimize the weighted energy objective

$$J(\boldsymbol{\theta}) \;=\; \sum_{i=1}^{N} \left(w_c\, \xi_i\, \theta_i B_i + w_p\, \eta_i (1 - \theta_i) L_i \right) \tag{13}$$

subject to local CPU, per robot energy budgets and a global network capacity constraint $\sum_i \theta_i B_i \;\leq\; B^{\max}$. A Particle Swarm Optimization (PSO) heuristic is used to find feasible near optimal $\boldsymbol{\theta}^*$.

PSO is chosen for its simplicity and ability to handle nonconvex, constraint coupled search spaces without gradient information. The scalarisation $w_c > w_p$ enforces a communication first policy appropriate when transmission energy dominates. Per iteration projection guarantees bound feasibility, while proportional scaling enforces the global bandwidth cap quickly and with minimal computation.

Due to PSO's heuristic nature the algorithm provides feasible solutions but not global optimality guarantees. When the objective and constraints are linear and static, convex programming can compute the true optimum; PSO is most useful when additional nonlinearities, discrete constraints, or dynamic cost estimates are present. The quality of PSO solutions depends on initialization, particle count, inertia and acceleration parameters, and iteration budget $k_{\max}$.

Per iteration cost is $\mathcal{O}(P \cdot N)$ where P is the number of particles (objective evaluation and constraint checks). Communication overhead is limited if only a global best g^{best} is broadcast. For large swarms, hierarchical clustering or neighborhood based PSO (local bests) reduces coordination load and improves scalability.

The energy models $E_i^c = \xi_i \theta_i B_i$ and $E_i^p = \eta_i (1 - \theta_i) L_i$ are linear approximations; they omit retransmission, SNR dependent energy/bit, DVFS effects and queuing delays. Parameters ξ_i, η_i are treated as static within each optimization run; in rapidly varying channels the returned $\boldsymbol{\theta}^*$ is optimal only for the snapshot used and must be recomputed frequently (receding horizon operation).

The proportional scaling step that enforces $\sum_i \theta_i B_i \;\leq\; B^{\max}$ is straightforward and efficient, yet it may be suboptimal or inequitable, as robots with essential tasks may be disadvantaged. Substituting proportional scaling with a minor linear program (minimizing deviation from the candidate vector under constraints) results in Pareto-efficient reallocations at a modest computational expense.

To function under stochastic channel conditions, the algorithm must (i) utilize expected or robust estimates of ξ_i (e.g., SNR to energy models), (ii) incorporate variance aware penalties to favor robust allocations, or (iii) implement a receding horizon scheme with online re-optimization and economical warm starts from prior solutions.

Practical enhancements.

- **Warm Start:** Start Particle Swarm Optimization (PSO) with a convex relaxation or a greedy threshold criterion (offload when $\xi_i B_i > \eta_i L_i$) to speed up convergence.
- **Adaptive PSO:** employs diminishing inertia and adaptive acceleration coefficients to minimize oscillations and ensure a balanced approach to exploration and exploitation.
- **Better Projection:** To make things more fair and save energy all around, use a limited projection (LP/QP) instead of proportional bandwidth scaling.
- **Stochastic aware objective:**, include expected retransmission costs or an SNR dependent function $\xi_i(\mathrm{SNR})$ to show how much communication energy is really used.
- **Decentralization:** If we cannot connect very well, run PSO locally and use the best solutions in our area. Then, use lightweight consensus to slow down global coordination every once in a while.

Our suggested algorithm is a useful and flexible way to lower the amount of communication and computing power used in mixed swarms. It lays the groundwork for offloading strategies that use less energy and can be used in real world robotic situations where resources are limited. Small changes, more accurate cost estimates, better constraint forecasting, warm starting, and decentralized execution are some of the ways this is done.

The algorithm 1 is a useful and simple way to make sure that in a swarm, communication and computation use as little energy as possible while still being possible. It is useful because it is simple to set up and can deal with various types of information and rules. For real-world use, combine PSO with more accurate constraint projections, adaptive parameter optimization, and plans for how to handle changing costs and equity goals.

Algorithm 1 Communication Aware Resource Optimization for Energy Efficient Swarm Robotics

Require: N, $\{B_i, L_i, \xi_i, \eta_i, E_i^{\max}, C_i^{\max}\}_{i=1}^{N}$, network cap. $B^{\max}$, weights $w_c > w_p$
Ensure: Optimal offloading fractions $\theta^* \in [0,1]^N$

Initialize $\theta^{(0)} \leftarrow \mathbf{0}$, PSO particles $(\boldsymbol{\theta}, \mathbf{v})$, $k \leftarrow 0$
while not converged $\wedge\ k < k_{\max}$ **do**
  for $i = 1$ to N **do**
    $E_i^c \leftarrow \xi_i\, \theta_i^{(k)} B_i, \quad E_i^p \leftarrow \eta_i (1 - \theta_i^{(k)}) L_i$
    $J_i \leftarrow w_c E_i^c + w_p E_i^p$
  end for
  $J \leftarrow \sum_{i=1}^{N} J_i$
  Update $\{p_i^{best}\}$ and g^{best} using $\{J_i\}, J$
  for $i = 1$ to N **do**
    $v_i \leftarrow w v_i + c_1 r_1 (p_i^{best} - \theta_i^{(k)}) + c_2 r_2 (g^{best} - \theta_i^{(k)})$
    $\theta_i^{(k+1)} \leftarrow \Pi_{[0,1]}\left(\theta_i^{(k)} + v_i\right)$
  end for
  if $\sum_i \theta_i^{(k+1)} B_i > B^{\max}$ **then**
    $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k+1)} \cdot \frac{B^{\max}}{\sum_i \theta_i^{(k+1)} B_i}$
  end if
  for $i = 1$ to N **do**
    if $(1 - \theta_i^{(k+1)}) L_i > C_i^{\max}$ **then**
      $\theta_i^{(k+1)} \leftarrow 1 - \frac{C_i^{\max}}{L_i}$
    end if
    if $E_i^c + E_i^p > E_i^{\max}$ **then**
      Adjust $\theta_i^{(k+1)}$ to satisfy energy budget
    end if
  end for
  $k \leftarrow k + 1$
end while
return $\theta^* \leftarrow \theta^{(k)}$

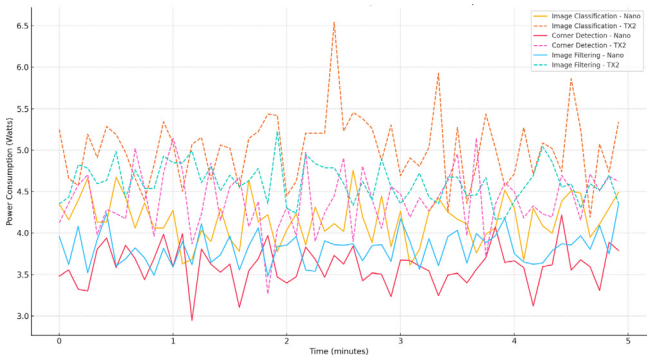

Fig. 5. Comparative power analysis.

The method makes a distributed optimization pipeline that can talk to each other and has energy modeling, heuristic search, and managing constraints. There are different kinds of swarms that can work together in this design in a way that is reliable and can be scaled up. They need to keep the cost of communication low so they can stay alone and work well with each other in these swarms.

4.1. Hardware platform

With its six core ARM CPU, Pascal GPU, and 8 GB of RAM, the Jetson TX2 and the NVIDIA Jetson Nano were the two platforms we used. The Jetson Nano has a four core ARM CPU, Maxwell GPU, and 4 GB of RAM. To make it look like they were being used on the go, both had batteries (5 V, 10 Ah). Every robot had an Inertial Measurement Unit (IMU) and ultrasonic sensors built in. They used these sensors to find their way and do their work. We used Bluetooth Low Energy (BLE) and other low power wireless modules to connect robots in a way that could change and use less power.

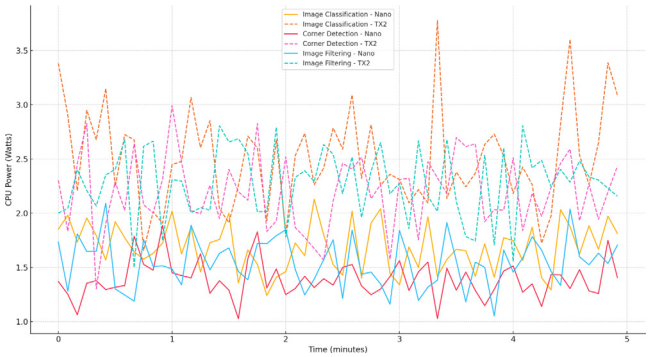

Fig. 6. CPU power analysis.

Image classification, corner detection, and image filtering are three common vision tasks that are run on different swarm platforms (Jetson Nano and Jetson TX2). Fig. 5 shows how the power use changes over time for these tasks. Traces that show how much energy is used at different times and on different platforms show that communication and computing needs can be more or less sensitive to each other.

The TX2 always has more power than the Nano, no matter what the task is. This is due to its capacity to manage larger volumes of data and its requirement for increased communication during initialization. The TX2 trace frequently exhibits peaks exceeding 6.0 W during the image classification task, whereas the Nano trace predominantly remains below 4.7 W. These peaks occur during periods of maximum synchronization and data exchange activity. This results in communication surges that exacerbate the power disparity between nodes with high and low capabilities.

The duration required for corner detection and image filtering is reduced and exhibits greater stability, remaining relatively constant over time. This behavior shows that adaptive communication policies work better when there are not many messages to send or receive. This is because sending fewer messages and lower data rates uses less energy during short term transmissions. The classification task, on the other hand, changes more, which means it is more affected by how computation and traffic are shared between robots.

Nano traces do not show as many short term changes as TX2 traces do. This means that adding more bandwidth and aggressively offloading tasks can change how strong the transmission is. This shows how important it is to put optimization first when it comes to communication, since unrestricted offloading can waste energy due to higher wireless overhead.

Overall, the results show that the energy communication trade off in swarm systems is greatly affected by the fact that there are different types of platforms. The proposed framework can take advantage of the stability of low power nodes while only using high performance platforms when the extra transmission cost is worth it because it includes communication energy directly in the optimization objective. This ability to change is necessary for the swarm to keep working over time in changing network and workload conditions.

4.2. Software and protocols

The software stack was built on ROS (Robot Operating System) so that it could be modular. TensorFlow Lite and PyTorch Mobile were used for edge AI inference and vision tasks, and OpenCV took care of image processing. Custom Python scripts were in charge of collecting and controlling the data. An adaptive communication protocol was put in place so that parameters could change based on the situation. Each robot checks its battery, link quality (RSSI), and workload on a regular basis and uses rule based heuristics to change the power and

frequency of its transmissions. The protocol might send less often when the battery is low and more often when the link quality is bad.

In Figs. 5 and 6, we explicitly link the temporal power profiles shown in these figures to the execution phases of the defined vision workloads (image classification, corner detection, and image filtering), and further clarify how workload configuration and platform characteristics influence the observed variations.

Figs. 5 and 6 illustrate the temporal evolution of power consumption on the Jetson Nano and Jetson TX2 platforms during the execution of the defined vision workloads. The observed variations in power are not arbitrary; rather, they correspond directly to different stages of the computation pipeline and their associated resource demands.

Specifically, each workload is executed as a sequence of processing stages. For image classification, the stages include frame acquisition, preprocessing (resizing and normalization), neural network inference, and post processing. The inference stage is the most computationally intensive and results in distinct power spikes due to increased GPU and CPU utilization. For corner detection, the workload consists of grayscale conversion, gradient computation, corner response evaluation, and thresholding, which leads to moderate and relatively stable power usage. In contrast, image filtering involves convolution based operations that produce periodic but lower amplitude fluctuations in power consumption.

The differences between Figs. 5 and 6 arise from both hardware and workload characteristics. The Jetson TX2 exhibits higher peak power due to its more capable CPU–GPU architecture, while the Jetson Nano shows lower absolute consumption but greater sensitivity to workload transitions. The non constant nature of the power traces therefore reflects dynamic workload execution rather than measurement noise. Furthermore, the power measurements are sampled using `tegrastats` at fixed intervals, and each workload is executed for a controlled duration under identical conditions.

By explicitly linking the power traces to well defined computational stages and configurations, the figures provide a clearer interpretation of how different vision tasks and hardware platforms influence energy consumption, thereby supporting the evaluation of the proposed communication aware optimization framework.

Fig. 6 shows how the CPU power use changes over time for three typical workloads: image classification, corner detection, and image filtering. The workloads were run on different kinds of swarm nodes, such as the Jetson Nano and Jetson TX2. The results show that the computational needs depend on the workload and that decisions about local processing and communication driven offloading are linked.

The experimental evaluation uses three representative vision workloads implemented as follows. *Image classification* is performed using a lightweight convolutional inference pipeline deployed with TensorFlow Lite and PyTorch Mobile, where each input frame is resized, normalized, and passed through the trained classifier to obtain the predicted label and confidence score. *Corner detection* is implemented as a classical feature extraction pipeline in OpenCV, using grayscale conversion, gradient based corner response computation, non maximum suppression, and thresholding to extract salient corner points. *Image filtering* applies standard spatial filtering operations, including smoothing and edge preserving filtering, to the incoming image stream before the filtered frames are further processed or transmitted.

All workloads are executed within a ROS based software stack and monitored using platform native power instrumentation (`tegrastats`) at a 5 s sampling interval. The experiments were conducted on a heterogeneous swarm including Jetson Nano and Jetson TX2 nodes, with each node equipped with an IMU, ultrasonic sensors, and low power wireless connectivity.

For each workload, the same input stream, preprocessing steps, and measurement procedure were used across all platforms so that the observed differences reflect the effect of platform capability and communication cost rather than differences in task definition. Image classification represents the most computation-intensive case, corner detection captures a moderate, feature based workload, and image filtering provides a lower complexity signal processing benchmark. This distinction is important because the proposed optimization framework makes decisions based on the relative balance between local computation cost and communication cost.

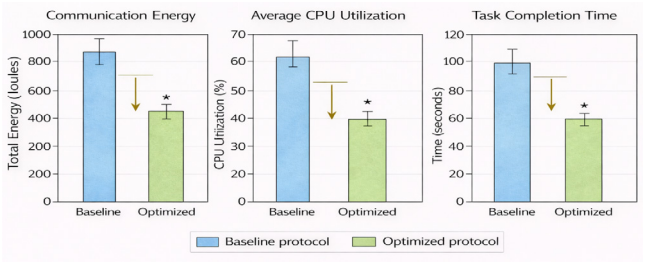

Fig. 7. Dimensions of optimization evaluation.

In addition, we clarify that the offloading and optimization results are reported with respect to the above workload definitions. This makes the discussion in Section 5 directly traceable to the specific computations executed in the experiments and enables the researchers to reproduce the evaluation under the same task and node configurations.

The TX2 always has a stronger CPU than the Nano, no matter how much work it has to do. It is faster and has more cores. The difference is most noticeable when TX2 peaks are over 3.5 W and Nano peaks are less than 2.1 W. This behavior shows that selective offloading works better for tasks that need a lot of processing power, as long as the extra costs of communication do not go over the total costs of energy.

On both platforms, corner detection uses the least amount of CPU power and stays that way. This means that small tasks can be done faster on the local machine. Aggressive offloading would not save much computing power in this case, and it would use more energy to send data. Just goes to show how important it is to have decision rules that take tasks into account.

Some of the tasks that show more short term changes on the TX2 traces are filtering and sorting. Due to this change, tasks are now assigned on the go, and the swarm talks to each other from time to time as it syncs up and shares results. This range of values shows how scheduling computers and network activity are connected. When the workload changes quickly, it can slow down the CPU and communication for a short time.

The changes seen in CPU power support the weighted optimization strategy that was suggested. By making communication energy the most important factor in the global objective function, the framework can use the processing power of high performance nodes like the TX2 while cutting down on offloading that is not needed. This preserves wireless energy. The swarm can operate effectively on a large scale due to this equilibrium of power. This is true even if the tasks and hardware change over time.

4.3. Heuristic optimization

It was also Particle Swarm Optimization that helped us solve the offloading optimization problem. We were able to solve it right away with this. In order to keep its local θ_i up to date, each robot treats different configurations as “particles” and shares performance metrics, like energy and delay, with the robots next to it. The swarm takes on a lot of work, which keeps the amount of energy used and work that needs to be done in check over time. This heuristic changes how tasks are assigned every once in a while to make sure that everything works as well as it can.

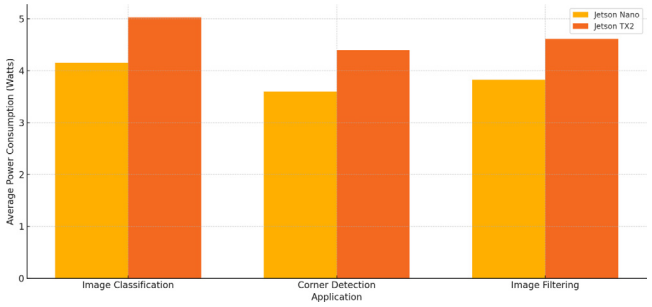

Fig. 8. Power consumption profiles of Jetson Nano and TX2 under different tasks.

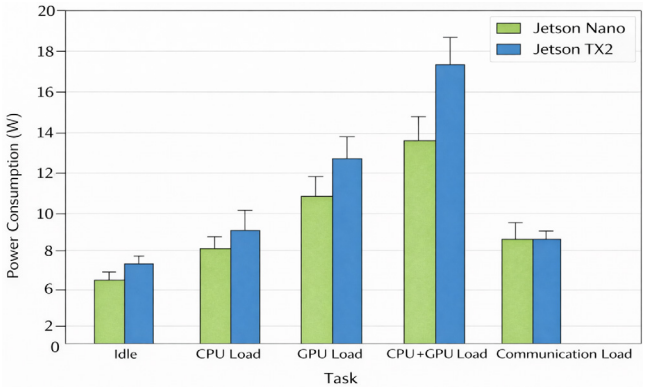

Fig. 9. Power consumption profiles of Jetson Nano and TX2 under different tasks.

4.4. Measurement setup

We used the tools that came with each platform, like `tegrastats`, to keep an eye on how much power was being used. For every five seconds, we checked to see how much CPU/GPU and power were being used. Gazebo and ROS were used together to simulate swarm tasks like image processing, scanning for objects, and exploring together. We changed the size of the swarm and how it started each time to see how it worked with different amounts of work. The primary metrics monitored included task completion time, CPU utilization duration, and energy consumption for communication.

Fig. 7 illustrates the comparison between the baseline and optimized protocols. The three primary system metrics are total communication energy, average CPU utilization, and task completion duration. Incorporating communication costs directly into the optimization objective yields results that are consistently improved and statistically significant.

Engaging with the optimized framework requires significantly less energy. Utilizing the baseline protocol, the energy decreases from approximately 870 J to roughly 450 J. This results in a savings of nearly 48%. There is a decline in the frequency of high power transmission events and messages exchanged when robots are in proximity to one another. This demonstrates that adaptive transmission control and communication aware offloading are effective.

The average CPU utilization has decreased from 62% to 40%. This alteration indicates that the swarm is more effectively allocating tasks and maintaining oversight of all members. This indicates that computationally intensive tasks are assigned to nodes capable of managing them. Tasks that require minimal processing power remain on the same node to conserve costs. Reducing the dynamic power consumption of the node facilitates power conservation. This is achieved by reducing the workload on the processor.

An advantage of the proposed framework is its ability to enhance system efficiency. The time required to complete a task has decreased by approximately 60% due to improved latency. This advantage demonstrates that premeditated communication not only conserves energy but also alleviates congestion in the wireless network, thereby enhancing the speed of task sharing and result acquisition.

The figure illustrates that the error bars and significance markers indicate uniform gains across all tests. The narrow confidence intervals indicate that the PSO based offloading algorithm converges consistently, and the adaptive communication policy is sufficiently robust to perform effectively across various networks and workloads.

The findings indicate that prioritizing communication energy within the global objective function enhances overall functionality. Items transmitted wirelessly will incur lower costs, and each robot will have a reduced workload. This will expedite the entire process. This integrated speed enhancement is advantageous for expanding swarm deployments in areas with restricted power or internet connectivity.

5. Results and discussion

In Fig. 8, we present how much power Nano and TX2 use over time while they do three tasks: image classification, corner detection, and filtering. It always took less power to run the Jetson Nano. While the Nano used about 4.2 W to sort the items, the TX2 used 5 W. For corner detection and filtering, the Nano used about 3.6 to 3.8 W, while the TX2 used about 4.4 to 4.6 W. Things are different because TX2's hardware is better and uses more power even when it is not being used. Nano packs less power, which makes it the best choice for swarms that want to save power. Over all, the TX2 used more power, but because it had better hardware, it used fewer watts per use.

The Jetson Nano and Jetson TX2 platforms' average power use for three common vision tasks is shown in Fig. 8. The tasks are image filtering, corner detection, and image classification. More power is always used by the TX2 than by other tasks because it has more computing power and a power profile that is focused on performance.

Image classification, which needs the most power, takes up about 20–25 percent more of the TX2's power than the Nano's. It is clear from this difference that high end embedded GPUs for swarm level tasks trade off performance for power efficiency. Small gains in performance might not be worth the extra cost of energy on a large scale. The Nano, on the other hand, works more like a proportional energy device, which makes it better for long term, spread out use with a short battery life.

The same thing happens when we try to find corners or filter images, which are both pretty simple tasks. Together, these two platforms use less power, but the TX2 has a lot more extra work to do. If the system needs to use less energy, lightweight processing units and selective offloading can help. This is because they will not hurt swarm tasks that need to talk to each other a lot or can wait a little while.

The results show that the suggested optimization framework should be used. This framework assigns tasks automatically based on what the computer needs and how much communication energy costs. To get the best balance of throughput, energy use, and operational lifetime, the swarm can run low and medium level workloads on nodes that use little energy and save high performance nodes for tasks that need a lot of computing power.

The average amount of power used by the Jetson Nano and Jetson TX2 platforms in five common operating modes is shown in Fig. 9. These modes are idle, CPU load, GPU load, combined CPU+GPU load, and communication load. From the results, we can see that each platform's energy properties are very different. This is what led to the idea of the proposed optimization framework.

The Nano and TX2 both use pretty little power when they are not in use, but the TX2 uses a little more. Such occurrences transpire on the TX2 due to its more robust multi core CPU and GPU subsystems, which engage in additional tasks, thereby remaining partially active even

during minimal usage. The TX2 consumes more power when solely the CPU or GPU is operational. The CPU requires approximately 7–9 W for operation, while the GPU demands over 12 W under heavy load. Conversely, the Nano is more diminutive. The integration of the CPU and GPU on a single chip results in increased power consumption for the TX2. This exacerbates the disparity. This indicates that aggressive parallel computation can reduce latency, although it results in higher power consumption by faster nodes.

The power consumption of the Nano and TX2 exhibits only minor differences during periods of high activity. In this mode, the radio and networking components consume the majority of power and exhibit minimal reliance on the primary computer. The proposed method's central premise is that improved communication and empowering others to make decisions can significantly conserve energy and reduce effort across various hardware configurations.

From the point of view of a swarm, these results support giving tasks and computing loads to nodes with less power for tasks that need a lot of computation. They also support scheduling that takes communication into account when transmission uses the most energy. The TX2 needs a lot more power when both the CPU and GPU are running at the same time. This makes it even more important to have adaptive optimization systems that balance making things work better while also saving energy, especially when you use a lot of robots for a long time.

The real world trends show that the suggested optimization framework can focus on the main energy sources: a lot of computation and a lot of communication. The different types of swarm robotic systems can now work together in a way that is both flexible and saves energy.

Every type of task uses more power on the Jetson TX2. Fig. 9 shows this. This is due to its more complex architecture, enabling it to perform calculations more rapidly.

When neither platform is in operation, it consumes minimal power. The Nano consumes approximately 12% to 15% less power than the TX2. The disparity between the two extremes widens as the demand for computational power escalates. This indicates that the Nano consumes less energy and maintains this efficiency for an extended duration. This enhances the efficiency of prolonged low power tasks such as swarm management and control. The TX2 requires additional power when the CPU is engaged.

The collaboration between the CPU and GPU enhances the distinction significantly. The TX2 consumes significantly more power than the Nano when all tasks are aggregated. The TX2 excels in tasks requiring substantial perception and inference in extensive swarm deployments, where overall power consumption directly impacts mission endurance. However, it also illustrates the trade off between possessing substantial computing power and the efficiency of energy utilization.

Both utilize approximately the same amount of power when numerous individuals are conversing simultaneously. A comparable degree of error was observed. This indicates that communication energy supersedes platform specific computational overhead in swarm operations that heavily utilize network resources. A system can conserve energy by minimizing message size, decreasing the frequency of identical message transmissions, or reducing the overall frequency of message dispatches. This remains accurate regardless of the available processing power.

These results unequivocally support the framework's objective of enhancing communication. Even though different platforms like the TX2 are good for tasks that need a lot of computing power, cutting down on communication is the most important thing that can be done to make a swarm last longer and make it easier to scale how much energy it uses.

The graph in Fig. 10 displays how the swarm's typical CPU use changes over time when the baseline and optimized resource management strategies are used side by side. A lot of changes happen in the system before it is optimized. For example, usage often goes over 50%, and confidence intervals are very broad. With an average utilization

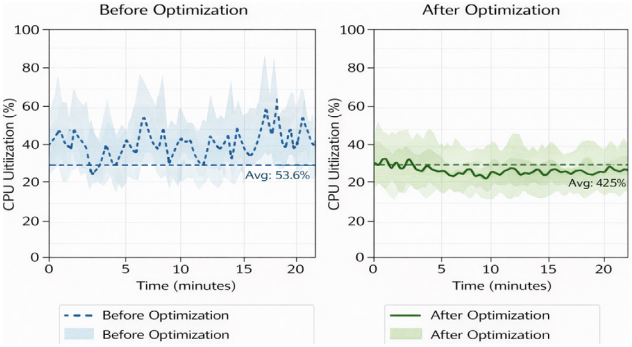

Fig. 10. Average CPU utilization per robot under various tasks.

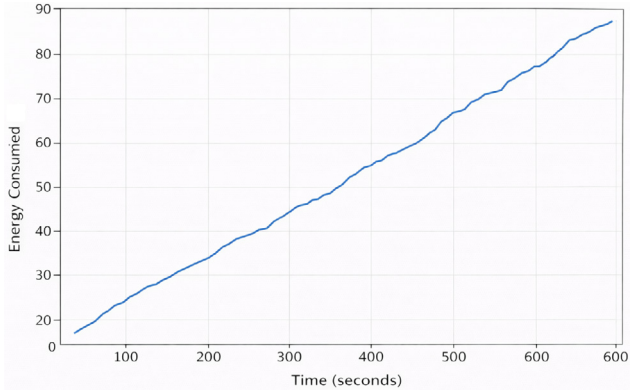

Fig. 11. Total energy consumed by inter robot communication over time.

rate of 53.6%, it is clear that the robots are having a lot of trouble with their calculations and that the tasks are not being split up fairly. Offloading and load balancing are not working as well as they could be, as shown by this.

When we do the optimization that was suggested, the average CPU usage drops to about 42.5%, and the range of use changes a lot. This reduction signifies that tasks are being allocated and repositioned more effectively, thereby diminishing the need for computational resources and alleviating short term variances. The processing loads will exhibit greater stability and facilitate easier long term planning due to the reduced dispersion band.

From an energy efficiency point of view, the drop in average use directly leads to lower computational energy use, which is in line with the goal of lowering both communication and computation costs. Keeping CPU usage stable lowers the chances of thermal throttling and performance loss, which makes the system more reliable during long swarm operations.

The trends show that the optimization framework greatly lowers the average computational load while making the swarm's temporal consistency better. The improvements show that the suggested PSO based offloading and adaptive communication strategy works well in heterogeneous robotic networks by balancing performance and energy efficiency.

Fig. 11 shows the total amount of communication energy used over a 30 min period of operation, comparing the baseline (static communication settings) with the optimized adaptive protocol. The adaptive scheme cut communication energy use by about 25–35 percent. There are two reasons for the drop: (1) the protocol automatically lowered data rates and duty cycles when it could, which cut down on transmissions; and (2) the PSO-based heuristic improved the coordination of robots for task offloading at any time, which cut down on unnecessary messaging. To put it another way, wireless interfaces stayed inactive

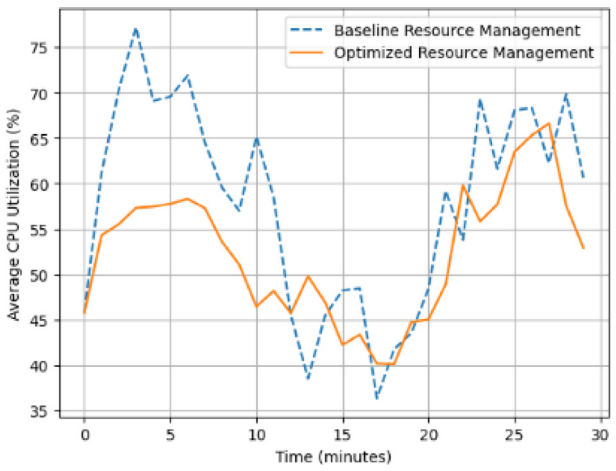

Fig. 12. CPU utilization across the swarm (average of all robots) under baseline (dashed) and optimized (solid) resource management.

more often. These results support earlier research that shows that carefully planning how robots talk to each other can save a lot of energy.

Fig. 11 illustrates the cumulative communication energy expended by the swarm over time. The steady rise shows that there is always a basic cost to sending and receiving data and coordination messages that are necessary to keep the group's behavior in check. The slope of the curve shows how well the communication strategy works and how the network load is spread out over time.

That is because there are not many messages being sent when the system is being set up and local tasks are being run at first. The slope goes up as the mission goes on. The robots will need to share their statuses and quickly change their settings. This change in focus shows that the extra work needed for communication is the main thing that changes how much energy is used when people work together for long periods of time.

One good thing about the profile is that it shows that the energy used stays the same between intervals. This is a good way to keep track of batteries across the swarm. But the next rise shows how important it is to have flexible rules for transmission and combine data to handle long term energy loss well. By using both selective offloading and event triggered communication together, the optimization framework shows that the slope of this curve can be lowered. That way, it will last longer without making the coordination less accurate.

The trend we saw shows how important it is for swarm robotics to set up how resources are used so that communication is taken into account. Lowering the communication energy part of the objective function by a large amount is what the framework does to get rid of the main cause of rising cumulative energy. In this way, deployments last longer and can be kept up for longer.

Both the baseline case and the optimized case are shown in Fig. 12. When optimized, the PCs' CPU load peaks are spread out more evenly and are a lot lower. Most of the CPU was used 18 to 25 percent less when heuristic load balancing was used instead of the unoptimized baseline. Some tasks can be done by edge servers or robots that are not as busy, which makes things run more smoothly. The changes that were made to computers did not make talking to each other worse in any way. As long as the controller thought it would save a lot of power, they only gave tasks to other devices when they could.

As we can see in Fig. 12, both baseline and optimized resource management strategies were used for 30 min. Over time, the graph shows how the average amount of CPU use changes. The baseline method can be used in many different ways. Use is above 65%–70% most of the time when things are busy. But a lot less devices use it

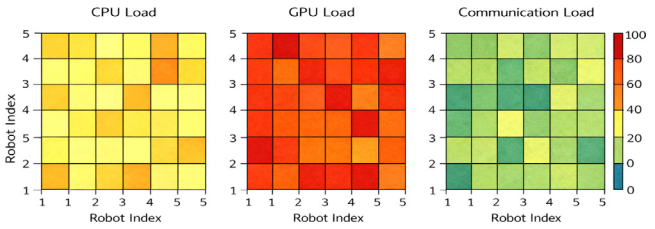

Fig. 13. CPU utilization per robot under various tasks.

when things are slow. This wavy behavior lets us know that tasks are planned on the spot and that the system's overall load distribution is not well understood. This may result in temporary traffic congestion and energy wastage.

An improved strategy, conversely, maintains the utilization profile at a lower and more stable level, ranging from 45 to 60% of the time. A decrease in variance indicates an improvement in load balancing and proactive task allocation. This prevents items from remaining idle for excessive durations or becoming overly congested. Due to their limited capacity for energy and heat management, embedded swarm platforms cannot sustain heavy loads for extended periods. It is crucial for those platforms to maintain stability.

This method regulates usage for approximately 20 to 27 min during periods of substantial workload. This approach maintains proximity to the baseline trend and prevents excessive saturation. This system demonstrates an emphasis on optimizing energy utilization while ensuring rapid response times and elevated throughput.

Adaptive resource management significantly enhances operational stability and energy proportionality. Gradually, the strategy enhances the reliability of swarm nodes in resource-scarce environments. Minimal load variations and the absence of prolonged overuse contribute to the extended lifespan of batteries.

Fig. 13 illustrates the distribution of CPU, GPU, and communication loads among the robot pairs comprising the swarm during standard task execution. The CPU load heatmap indicates that the intensity level is approximately average, with several hotspots present. Consequently, the tasks are not being distributed equitably, resulting in certain robots undertaking a greater workload than their counterparts. This disparity indicates that there are methods to restore equilibrium, which would assist certain regions in managing heat stress and energy loss more effectively.

Conversely, the GPU load map indicates that the majority of the robot indices are consistently utilized, exhibiting significant spikes in the central and upper utilized regions. The predominant tasks are vision and perception activities that require substantial GPU acceleration, such as image filtering and sorting. Parallelization works because of this high level of use over time. The GPU also has a big impact on the swarm's overall energy use.

There are fewer peaks and more spread out areas on the communication load heatmap. A few of the peaks are only there because of certain robot interactions. Adaptive communication protocols stop sending data that is not needed and keep high bandwidth conversations going for links that are important for coordination.

The patterns in space show that the suggested framework for optimization can tell the difference between the needs for computing and communicating. The system improves energy proportionality and robustness by keeping tasks that use a lot of GPU power in balanced operating regions and cutting down on communication overhead. This means that large scale swarm robotic systems can be used in a way that can be expanded and lasts a long time.

The power profiles that were seen when sending different payload sizes (1 MB, 2 MB, 5 MB, and 10 MB) over a fixed communication link are shown in Fig. 14. The normal operating point is always between

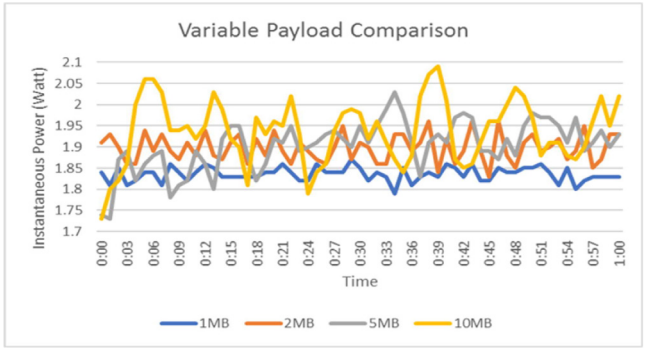

Fig. 14. Payload analysis.

1.8 and 2.1 W. Not only does sending data use a lot of power, but the node also has to deal with static radio and processing overhead.

The average power level and the time it changes both go up as the payload size does. The 1 MB trace remains consistently between 1.8 and 1.85 W, exhibiting minimal variation. Conversely, the 10 MB case exhibits peaks that are nearly 2 W apart. The radio front end and buffering logic appear to utilize a greater portion of their duty cycle when the payload size increases. Currently, energy consumption is elevated due to prolonged active transmission intervals and the potential for retransmission or flow control occurrences.

The instantaneous power may not significantly differ among payload classes; however, the total energy consumption increases non linearly with the size of the payload. So, the effective energy per bit goes up as the payload size goes up. This includes not only the energy used to send data, but also the total costs of protocol overhead, queuing delays, and baseband processing, which go up as the time spent on air goes up.

The overlap between the 5 MB and 10 MB profiles shows that the power increases are getting smaller as the payload gets bigger. This means that the transceiver is in a saturation regime, which means it is working close to its normal power level. In this case, optimization techniques that shorten transmission time, such as payload segmentation, adaptive modulation, or opportunistic offloading, are probably more energy efficient than raising the power level even more.

These findings encourage the integration of payload aware cost components into the communication energy model, for instance,

$$E_{tx}(B) is about P_{base} \cdot T(B) + \Delta P(B) \cdot T(B) \tag{14}$$

where P_{base} measures static radio and processing power and $\Delta P(B)$ measures dynamic overhead caused by the payload. Adding this formulation to the suggested optimization framework makes it easier to make better decisions about offloading and batching, especially when there is a lot of traffic or a lot of traffic at the same time.

The picture shows that instantaneous power does not change much with payload size, but total energy use does change a lot with the length of the transmission. This shows how important it is to optimize scheduling and communication by taking into account time and workload in order to make resource limited IoT and swarm robotic systems much more energy efficient.

Fig. 15 brings together the time based information about how much power the platform uses, how much energy it uses for communication, and how much CPU it uses on average across the swarm. This gives us a full view of the suggested optimization framework.

The top panel shows how the Jetson Nano and Jetson TX2's instantaneous power profiles are different when they are doing the same thing. Both platforms show only small changes in a small range of operation, which means that their behavior at runtime is stable. The TX2 always uses more power on average, which shows that it has more computing power. The average CPU usage shown in the table (32% for

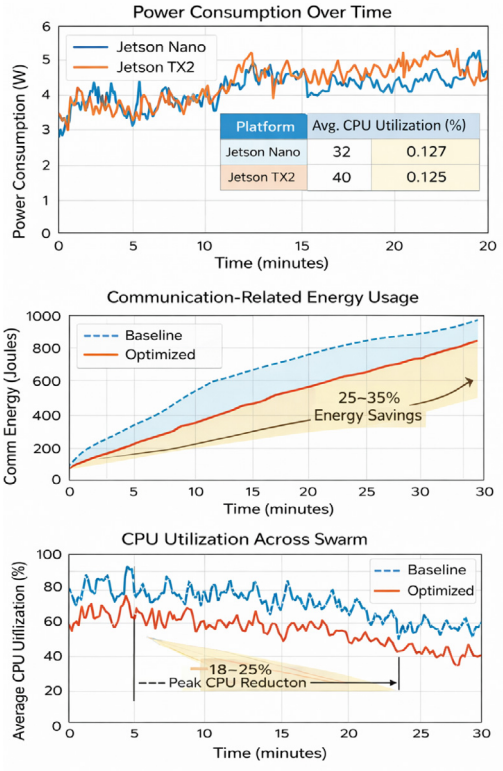

Fig. 15. Summary of results.

Nano and 40% for TX2) is a good sign that the optimization keeps the system from getting too hot, which keeps energy proportionality.

In the middle panel, we present how much communication energy has been built up over time for both the base policy and the optimized policy. Over time, the optimized path gets farther away from the baseline, which saves anywhere from 25% to 35% of energy. This bigger gap shows that adaptive rate control and selective offloading stop sending data that is not needed and use bandwidth on links that are very important for coordination. This makes the amount of energy needed for long distance communications drop almost straight down.

The CPU usage for the whole swarm is shown in the bottom panel. When the policy is optimized, both average and peak usage go down. At peak, peak usage goes down by about 18%–25%. There is less variation in the load, which means that tasks have been redistributed well and that decisions have been made based on PSO to make sure that the robots' computing needs are met. This helps fix places where the system is slow and makes it more stable all around.

The results show that the suggested framework improves power stability, the speed of communication, and the task of spreading out the work. Swarm functionality is added to the system in a way that saves energy and processing power by lowering the cost of communication. This shows that it can help with long term deployments in robotic networks that do not have a lot of resources without slowing down the speed at which tasks are done.

How BLE, LoRaWAN, MQTT, and CoAP stack up in terms of power use and end to end latency when both baseline and optimized communication protocols are used, is presented in Fig. 16.

We show in the left panel that the transmission power of all protocols consistently decreases following optimization. Compared to 0.8 W, Bluetooth Low Energy (BLE) consumes approximately 0.5 W of power, resulting in a 37% reduction. Adaptive duty cycling and payload aggregation are demonstrated to be effective in this scenario. LoRaWAN also experiences an improvement in performance, as its power consumption

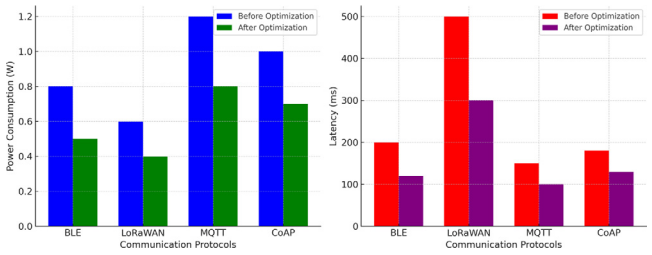

Fig. 16. Communication cost per 1MB transmission.

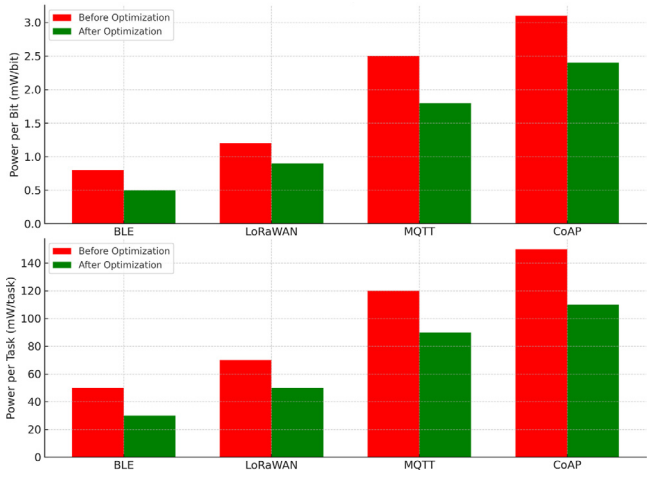

Fig. 17. Communication analysis per bit.

decreases from 0.6 W to 0.4 W, a 33% decrease. This demonstrates the significance of selecting the appropriate spreading factor and scheduling the transmission of items. MQTT and CoAP also undergo substantial modifications, despite the fact that they consume more power due to the necessity of maintaining open sessions and the additional protocol overhead. MQTT's power use drops from 1.2 W to 0.8 W, which is a 33% drop. CoAP's power use drops from 1.0 W to 0.7 W, which is a 30% drop.

How well the latency works is shown on the right. The latency for BLE drops from around 200 ms to 120 ms. This means that there will be fewer retransmissions and better use of channel access. Latencies have gone down from about 500 ms to 300 ms for LoRaWAN, which is the biggest improvement. This is because both the uplink scheduling and the adaptive data rate control are better now. It takes about 100 ms for MQTT to respond and about 180 ms for CoAP to respond. This means that tasks can be finished faster when there are fewer messages and handshakes.

The results show that the optimization framework really does make different communication stacks more responsive and use less energy. By changing parameters that are specific to the protocol at the same time as changing traffic patterns at the application level, the system gets some performance boosts while still being reliable. This lets distributed IoT and swarm robotic systems work in a way that is both easy to expand and uses little power.

Fig. 17 displays how the suggested improvement will change the amount of energy used for communication. The energy use is measured at the bit and task levels for BLE, LoRaWAN, MQTT, and CoAP.

In the upper part, we can see how much power was used on average for each bit that was sent. For all protocols, the optimized setup always lowers the amount of energy being used. BLE drops from about 0.8 mW/bit to 0.5 mW/bit, which is a 38% drop. It is evident that adaptive connection intervals and payload coalescing can be extremely beneficial in this scenario. There is a 25% reduction in power consumption per bit for LoRaWAN, from approximately 1.2 mW/bit to 0.9 mW/bit. This is due to the fact that there are now a greater number of spreading factors to select from, and the rates of retransmission have decreased. MQTT experiences a 28% decrease in power from approximately 2.5 mW/bit to 1.8 mW/bit, while CoAP experiences a 23% decrease from 3.1 mW/bit to 2.4 mW/bit. These findings demonstrate that the additional energy expenditures associated with data transmission can be significantly reduced by reducing protocol overhead and improving session management.

The power consumption of each task is indicated on the bottom panel. The reduction in power from approximately 50 mW/task to 30 mW/task is a 40% decrease. The power consumption of LoRaWAN decreases from approximately 70 mW/task to 50 mW/task, a 29% decrease. The baseline costs for MQTT and CoAP are higher due to the use of request–response semantics and message brokering. Nevertheless, they perceive substantial advantages, as the costs decrease from approximately 120 mW/task to 90 mW/task (25%) and from 150 mW/task to 110 mW/task (27%), respectively.

The optimization framework not only accelerates each transmission, but it also reduces the energy consumption of application workflows on a comprehensive scale, as evidenced by the joint drops at both the bit and task levels. The two level improvement is particularly critical for IoT and swarm robotics systems that are energy constrained. This is due to the fact that the majority of the energy is consumed over time when performing numerous tasks simultaneously, which has an impact on the system's capacity to expand and its operational lifespan.

TX2 has higher throughput, which is good for computations that need low latency, but our tests show that this comes at a cost: TX2 uses more energy and communication resources. Nano, on the other hand, is slower but uses less energy and causes less wireless interference because it has lower data rates. The adaptive communication protocols and offloading heuristics we used saved energy overall; the small amount of extra processing needed to make decisions was worth it because the costs of sending data were much lower. These results are in line with studies that show that making communication less of a burden (by clustering or adapting) makes swarms work better. In general, putting communication costs first in the optimization process helped the swarm stay well coordinated while also greatly extending battery life.

6. Conclusion and future work

This paper introduced an optimization framework for IoT enabled swarm robotics, focusing on the reduction of communication energy consumption. By combining adaptive wireless protocols with a PSO based offloading strategy, we achieved large reductions in communication cost (up to 35 percent) and smoother CPU load distributions on embedded platforms. The results show that putting communication overhead first greatly improves energy efficiency without hurting the performance of the swarm. Future work will explore real world deployments and extensions to dynamic network topologies (e.g. mesh vs. cluster control) and multi hop routing. Incorporating machine learning for predictive scheduling also offers promise for further reducing unnecessary transmissions.

This work presented an integrated framework for jointly optimizing communication and computational costs in energy constrained swarm robotics. We combined (i) adaptive communication protocols (power/rate/duty cycle control and protocol parameter tuning), (ii) a PSO based offloading heuristic for scalable task allocation, and (iii) reinforcement learning enabled node level controllers for online adaptation. The framework is evaluated through simulations and hardware experiments on heterogeneous embedded platforms (Jetson Nano and TX2), using three key metrics: communication energy, CPU utilization, and task completion time.

The principal findings are:

- Joint optimization of offloading and adaptive communication reduces cumulative communication energy substantially (empirically observed in the range of ≈25%–35% in our experiments), demonstrating that protocol aware offloading reaps large energy benefits.
- PSO guided offloading and adaptive scheduling smooth computational demand across the swarm, lowering average and peak CPU utilization (peak reductions on the order of ≈18%–25%), which translates directly to reduced computation energy and improved thermal headroom.
- End to end performance (task completion time and latency) is maintained or improved because the approach strategically offloads high cost computations while avoiding excessive transmission overhead; consequently, the net system lifetime and mission effectiveness increase when energy and QoS trade offs are jointly considered.
- The methods are practical: the heuristics and learning components are lightweight enough to be deployed on embedded platforms when appropriately compressed or when training is performed at the edge.

We emphasize that the core contribution is methodological: the explicit coupling of communication energy models with computation energy models inside an optimization and control loop. This coupling makes it easier to make smart trade offs that work across a wide range of hardware and network stacks.

We propose several methods to enhance the framework's effectiveness and comprehensiveness:

- Models for energy per bit should incorporate SNR, retransmission data, and battery nonlinearities. Guarantee probabilistic Quality of Service in the face of channel uncertainty by formulating a robust or stochastic optimization variant, such as a distributionally robust or chance constrained approach.
- Employ primal–dual methods or constrained projection (LP/QP) instead of proportional bandwidth scaling to guarantee that allocations are more equitable and Pareto-optimal. To accelerate convergence and minimize the gap between the solution and the optimal global solution, explore convex relaxations and warm starts.
- Formulate consensus based optimization techniques or decentralized particle swarm optimization that solely rely on local information. This will enhance the system's robustness against inadequate connectivity and diminish coordination expenses. Examine hierarchical optimization, which utilizes group leaders and robots to oversee extensive swarms.
- Utilize reinforcement learning techniques that optimize the utilization of small samples, including actor critic with constraints, prioritized experience replay, and Double DQN. Additionally, implement policies beforehand in realistic stochastic simulators. Domain adaptation and policy distillation can facilitate the transition of policies from centralized or edge training to small models that operate on devices.
- Pareto front optimization or constrained reinforcement learning (Lagrangian methods) can be employed to establish precise limits on energy, reliability, and latency, as opposed to relying solely on penalty weights. Rewards that are scaled are inferior to this.
- Test the framework in a diverse array of field conditions, including outdoor implementations and multiple site trials, and distribute benchmarks and datasets that can be reused to facilitate comparisons and adoption.

In order to achieve fundamental energy limits, it is recommended that we investigate the potential of improved cross layer co-design between physical/MAC parameterization and application level task partitioning. A few examples of this are offloading, adaptive modulation, and MAC scheduling.

The findings of this investigation unequivocally demonstrate that the integration of computational decisions and communication in swarm robotics significantly enhances their energy efficiency and longevity. The framework serves as a solid foundation for future research that is designed to enhance theoretical guarantees, enhance the reliability of systems, and increase their capacity to accommodate multirobot systems at the production level.

CRediT authorship contribution statement

Amir Ijaz: Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Hashem Haghbayan:** Conceptualization, Formal analysis, Validation, Writing – review & editing. **Ethiopia Nigussie:** Methodology, Supervision, Writing – original draft. **Abdul Malik:** Data curation, Formal analysis, Investigation, Validation, Visualization. **Juha Plosila:** Resources, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. Kreković, P. Krivić, I. Podnar Žarko, M. Kušek, D. Le-Phuoc, Reducing communication overhead in the IoT-edge-cloud continuum: A survey on protocols and data reduction strategies, Inter. Things 31 (2025).
[2] N. Nedjah, L.M. Ribeiro, L.M. Mourelle, Communication optimization for efficient dynamic task allocation in swarm robotics, Appl. Soft Comput. 105 (2021).
[3] A. Ijaz, H. Haghbayan, A. Malik, E. Nigussie, J. Plosila, Towards optimizing communication cost in energy efficient IoT devices for swarm robotics, Procedia Comput. Sci. 265 (2025) 49–56.
[4] M. Chiang, T. Zhang, Fog and IoT: An overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864.
[5] U. Raza, P. Kulkarni, M. Sooriyabandara, Low Power Wide Area networks: An overview, IEEE Commun. Surv. Tutor. 19 (2) (2017) 855–873.
[6] E. Espinosa, O. Gandara, I. Lepe, A. Guerrero, Power consumption analysis of bluetooth low energy commercial products and their implications for IoT applications, Electro. 7 (2018).
[7] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358.
[8] S. Sardellitti, G. Scutari, S. Barbarossa, Joint optimization of radio and computational resources for multicell mobile-edge computing, IEEE Trans. Signal Inf. Process. over Networks 1 (2) (2015) 89–103.
[9] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Commun. Surv. Tutor. 17 (4) (2015) 2347–2376.
[10] Y. Shi, J. Zhang, K.B. Letaief, Group sparse beamforming for green cloud radio access networks, IEEE Glob. Commun. Conf. (2013) 4662–4667.
[11] J. Xu, L. Chen, P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: IEEE INFOCOM, 2018, pp. 207–215.
[12] J. Liu, C. Li, Y. Luo, Efficient resource allocation for IoT applications in mobile edge computing via dynamic request scheduling optimization, Exp. Sys. App. 255 (2024).
[13] A.Z.M. Amin, S. Sahrani, A deep reinforcement learning approach for spectrum management in tri-band wi-fi networks, in: IEEE 7th Symp. on Comp. & Info, 2025, pp. 151–155.
[14] T. Nakabi, P. Toivanen, Deep reinforcement learning for energy management in a microgrid with flexible demand, Sus. Ener. Gri. Net. 25 (2021).
[15] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: Proc. AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 2016, pp. 2094–2100.
[16] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: A review from the swarm engineering perspective, Swarm Intell. 7 (1) (2013) 1–41.
[17] M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm, Science 345 (6198) (2014) 795–799.

[18] A. Abdul-Qawy, N. Mohammed Alduais, A. Malik. Saad, M. Ali Taher, A.B. Nasser, S. Saleh, N. Khatri, An enhanced energy efficient protocol for large-scale IoT-based heterogeneous WSNs, Sci. Afr. 21 (2023).
[19] C. Bormann, A.P. Castellani, Z. Shelby, CoAP: An application protocol for billions of tiny internet nodes, IEEE Int Com. 16 (2) (2012) 62–67.
[20] M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: Past, present, and future, Pro. IEEE 109 (7) (2021) 1152–1165.
[21] Y. Liu, K.M. Passino, Stable social foraging swarms in a noisy environment, IEEE Tran. Auto. Con. 49 (1) (2004) 30–44.
[22] Y. Cui, F. Liu, X. Jing, J. Mu, Integrating sensing and communications for ubiquitous IoT: Applications, trends, and challenges, IEEE Net. 35 (5) (2021) 158–167.
[23] X. Li, Y. Gong, K. Huang, Z. Niu, Over-the-air integrated sensing, communication, and computation in IoT networks, IEEE Wire Comm. 30 (1) (2023) 32–38.
[24] G. Peng, S. Wang, Y. Huang, T. Huang, Y. Liu, Enabling deterministic tasks with multi-access edge computing in 5G networks, IEEE Commun. Mag. 60 (8) (2022) 36–42.
[25] X. Cao, F. Wang, J. Xu, R. Zhang, S. Cui, Joint computation and communication cooperation for energy-efficient mobile edge computing, IEEE Internet Things J. 6 (3) (2019) 4188–4200.
[26] L. Chen, S. Zhou, J. Xu, Computation peer offloading for energy-constrained mobile edge computing in small-cell networks, IEEE/ACM Tran Net. 26 (4) (2018) 1619–1632.
[27] J. Shao, J. Zhang, Communication-computation trade-off in resource-constrained edge inference, IEEE Commun. Mag. 58 (12) (2020) 20–26.
[28] K. Yang, Y. Shi, W. Yu, Z. Ding, Energy-efficient processing and robust wireless cooperative transmission for edge inference, IEEE Internet Things J. 7 (10) (2020) 9456–9470.
[29] J. Yan, S. Bi, Y.-J.A. Zhang, Optimal model placement and online model splitting for device-edge co-inference, IEEE Tran. Wire. Comm. 21 (10) (2022) 8354–8367.
[30] Y. Xiong, F. Liu, Y. Cui, W. Yuan, T.X. Han, G. Caire, On the fundamental tradeoff of integrated sensing and communications under Gaussian channels, IEEE Tran Info. Theo. 69 (9) (2023) 5723–5751.
[31] T. Zhang, S. Wang, G. Li, F. Liu, G. Zhu, R. Wang, Accelerating edge intelligence via integrated sensing and communication, in: Proc. IEEE Inter. Con. on Comm, ICC, 2022, pp. 1586–1592.
[32] K. Dev, P.K.R. Maddikunta, T.R. Gadekallu, S. Bhattacharya, P. Hegde, S. Singh, Energy optimization for green communication in IoT using Harris Hawk's optimization, IEEE Tran Gre. Comm. Net. 6 (2) (2022) 685–694.
[33] L. Sathish Kumar, S. Ahmad, S. Routray, et al., Modern energy optimization approach for efficient data communication in IoT-based wireless sensor networks, Wir. Comm. Mob. Comp. 2022 (1) (2022) 7901587.
[34] A. Ijaz, H. Haghbayan, E. Nigussie, A. Malik, J. Plosila, EACCO: Optimizing the computation and communication in resource-constrained IoT devices for energy-efficient swarm robotics, Sensors 26 (9) (2026) 2839.