Renewable Energy and Power Quality Journal
https://repqj.com/index.php/repqj/index
RE&PQJ, Vol. 24, No. 5, 2026

Battery-Side Energy Efficiency of Wirelessly Connected Robots Using the Joint Offloading, Compression and Scaling Offloading and Scheduling Policy

Amir Ijaz[1,*], Hashem Haghbayan[1], Juha Plosila[1], Ethiopia Nigussie[2]

[1]Department of Computing, University of Turku, 20014 Turku, Finland

[2]Faculty of Technology, University of Turku, 20014 Turku, Finland

[*]Corresponding Author: Amir Ijaz, amir.ijaz@utu.fi

Article History

Received: 01 August 2026 Revised: 08 September 2026

Accepted: 29 October 2026 Available Online: 05 November 2026

Abstract. Battery capacity is the binding constraint on how long a fleet of mobile robots can work between charges. In wirelessly connected fleets, the robot battery is shared between three competing consumers: locomotion, onboard computation, and the radio. All energy figures in this paper refer exclusively to the robot battery; the edge server is mains-powered and outside the optimisation scope. Offloading perception to the edge server removes computation from the battery, but it consumes airtime on a shared wireless medium and a late result forces the robot to slow down, lengthening the mission and increasing locomotion energy drawn from the same cell. We develop a battery-side energy model that prices a missed perception deadline in joules through its effect on mission duration, derive the penalty in closed form as $\lambda = 5.18\,\mathrm{J}$, and propose joint offloading, compression and scaling (JOCS), a decentralised per-epoch policy that jointly selects the offloading decision, payload compression level, transmit power, and processor frequency for each pipeline stage. JOCS admits offload requests in decreasing order of battery energy saved per second of airtime consumed, subject to a feasibility test on the self-induced congestion. Simulated over 30 independent channel realisations, JOCS reduced robot battery energy by 22.6% relative to run-to-deadline local execution at eight robots per access point (95% CI: ±0.04%), with zero missed deadlines. At 24 robots, JOCS saved 19.5% against all-local and 31.9% against the best competing baseline (95% CI: ±0.7%). An extended ablation study with six component-removal variants isolates the contributions of compression, admission control, the latency-energy coupling term, and the robustness margin. A congestion-model sensitivity study with three alternative delay functions shows that the nominal saving is stable, while the congested-fleet advantage requires recalibration to the measured channel before deployment.

Key words. Mobile robots, Computation offloading, Battery energy efficiency, Edge computing, Wireless networks, admission control, resource allocation.

Abbreviations

AP: access point

DVFS: dynamic voltage and frequency scaling

JOCS: joint offloading, compression and scaling

MAC: medium access control

MCS: modulation and coding scheme

MEC: mobile edge computing

RTT: round-trip time

SLAM: simultaneous localisation and mapping

SNR: signal-to-noise ratio

SoC: system on chip

1. Introduction

Scope. All energy figures, contributions, and conclusions in this paper concern exclusively the robot's battery energy. Locomotion, onboard computation and radio transmission all draw on the same electrochemical cell; the edge server is mains-powered and lies outside the optimisation scope. This scope is declared at the outset because: (i) improving robot battery life may shift energy to the mains grid (discussed in Section 6.8); and (ii) the

results are relevant to the journal's energy-efficiency theme specifically in the context of battery-powered autonomous systems, whose charge cycle count is the binding operational constraint.

Motivation. The working life of a battery-powered mobile robot between charges is decided by a handful of subsystems that draw on the same cell: locomotion (the largest consumer), on-board computation (growing as robots adopt learned perception), and the radio (easy to overlook). Offloading heavy perception to a mains-powered edge server removes computation from the battery. But the accounting is not straightforward. A robot that cannot simply wait: if a perception result arrives after the control loop deadline, a safety-conscious robot slows down. A slower robot takes longer to finish its path, increasing locomotion energy from the same battery. Latency is itself a battery energy cost. The empirical study of Dordevic et al. [1] confirmed this tension on a TurtleBot3: offloading cut onboard energy but generated enough traffic to congest the WiFi link.

Contributions. We develop an energy model that places all three consumers in a single battery budget, derive the latency-energy coupling in closed form, and propose JOCS. Specifically:

- A battery-side energy model for a wirelessly connected robot fleet that prices a missed perception deadline in joules through its effect on mission duration (Section 3).

- JOCS, a decentralised policy with a formal pseudocode specification (Algorithm 1), that admits offload requests under a utilisation feasibility test and requires no coordination messages among robots (Section 4).

- An evaluation over 30 independent seeds for both $N = 8$ and $N = 24$ robots, with paired run-by-run confidence intervals, an extended six-variant ablation study, a congestion-model sensitivity analysis, and explicit discussion of simulation scope, parameter provenance, and transferability (Sections 5-6).

2. Related Work

Energy models for mobile robots. Mei et al. [2] characterised the power draw of a differential-drive platform and proposed the quadratic-plus-linear motion model we adopt. Wu et al. [3] extended this to battery-aware path planning, reporting up to 22.87% battery savings by re-selecting paths online. Naseri et al. [4] use reinforcement learning to jointly manage computational and mechanical energy, sharing our insight that computation and locomotion draw on the same battery. Their setting is a single robot without a shared wireless medium; combining their learned velocity planner with JOCS-style offloading is a natural direction for future work.

Computation offloading and MEC. Kumar and Lu [5] established the computation-to-communication trade-off in mobile offloading. MAUI [6] demonstrated practical code offload on smartphones. The MEC literature formalised joint allocation of radio and computational resources [7,8], including partial off-loading with DVFS [9] and distributed offloading on shared channels [10]. Zhang and Wen [11] treat applications as general task topologies. Liu et al. [12] study delay-optimal partial offloading under stochastic channel models, and Dinh et al. [13] survey off-loading frameworks for heterogeneous networks. Ren et al. [14] address importance-aware scheduling for wireless MEC, showing that channel-quality-aware admission improves throughput; our admission criterion additionally prices the locomotion consequence of latency. None of these works couples latency back to mechanical energy: in a smartphone, waiting for an offloaded result costs no extra battery energy. In a mobile robot, a late result triggers a speed reduction that lengthens the mission, charging extra locomotion energy to the battery-a coupling absent from all prior MEC models. In our model, E_i^{off} contains only radio energy; the locomotion energy enters E_{tot} through the lateness-price term $\lambda \mathbb{1}\left\{T_{\text{exec}}^{(i)} > D_i\right\}$ in (12).

Contention-aware and multi-agent offloading. Spatharakis et al. [15] study task offloading and resource allocation for a mobile robot at a MEC server, showing that offloading latency is the dominant constraint; our work extends this to fleet-scale wireless congestion. Chen et al. [10] use a game-theoretic formulation for multi-user offloading on interference-limited channels; their NP-hardness result motivates our greedy approximation. Recent robotic edge computing surveys [7] identify contention as an open problem for fleets. Wang et al. [16] study admission control for MEC but do not model the locomotion-energy feedback. Zhou et al. [17] optimise computation allocation in vehicular fog computing under contention; their contract-matching approach requires a centralised server, whereas JOCS is fully decentralised. The novelty of JOCS relative to these works is the joint optimisation of the offloading decision, compression level, transmit power, and processor frequency, combined with a self-interference-aware admission criterion derived directly from the battery-latency coupling.

3. System Model

We consider K robots performing a coverage mission in an indoor area served by a single access point backed by an edge server. Time is divided into epochs of duration $\Delta t = 0.5\,\text{s}$; at each epoch every robot runs a perception and planning pipeline and executes one motion step. The system architecture is illustrated in Figure 1. Table 1 lists all symbols; Table 2 gives all numerical values with sources.

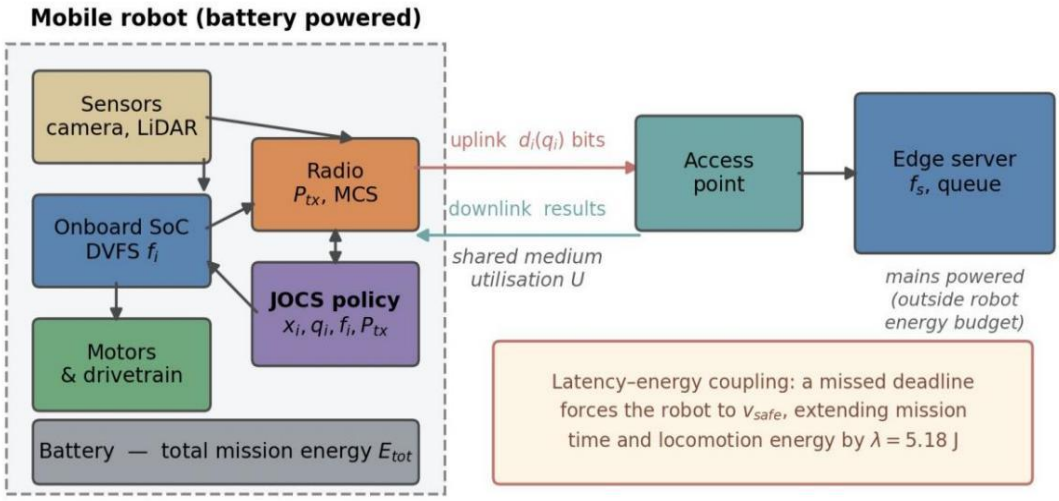


Figure 1. System architecture. The JOCS policy reads U_{t-1} from the radio driver and selects x_i, q_i, P_{tx}, f_i per pipeline stage. All energy optimised by JOCS is robot battery energy; the edge server is mains-powered and outside the optimisation scope (Section 6.8).

Table 1. Notation. All symbols used in the model and algorithm are listed here; t_q, J, E_{wake}, P_{rx}, η_{PA}, P_{ckt}, η_s, f_s, T_{RTT} appear in Eqs. (8)-(10) and are defined inline.

Symbol	Meaning
v_t	robot speed in epoch t (ms^{-1})
$P_{\text{mot}}(v)$	locomotion power (W)
P_{base}	platform and sensor idle power (W)
w_i	workload of task i (cycle)
D_i	deadline of task i (s)
f_i	processor frequency for task i (Hz)
κ	effective switched capacitance ($\text{Ws}^3/\text{cycle}^3$)
$q_i \in \mathcal{Q}$	compression level of task i
$d_i(q_i)$	payload size at level q_i (bits)
$a(q_i)$	detection accuracy at level q_i relative to $q = 0$
$x_i \in \{0,1\}$	offloading decision
P_{tx}	transmit power (W)
γ_k	uplink SNR of robot k (linear)
C_k	solo link goodput (bit s^{-1})
α_i	airtime of task i : $d_i(q_i)/C_k$ (s)
U	channel utilisation (busy fraction)
U_{bg}	background utilisation from non-robot stations
$\varphi(U)$	congestion delay factor
$\varphi_{\max}$	cap on φ
T_{to}	transport timeout (s)
$\mu \in (0,1)$	robustness margin
$\eta \geq 1$	Load-inflation factor
λ	battery energy price of one missed deadline (J)

Table 2. Complete parameter list, sources, and ±20% sensitivity. E_{ref} = JOCS battery energy at baseline ($N = 8$, mean over 30 seeds).

Parameter	Value	Source	±20% on E_{ref}
a, b, c	8.0, 12.0, 6.0	[2]	±1.5%
P_{base}	4.5 W	data sheet	±5.4%
v_{nom}	$0.40\,\mathrm{m\,s^{-1}}$	scenario	-
v_{safe}	$0.15\,\mathrm{m\,s^{-1}}$	scenario	-
Δt	0.5 s	scenario	-
κ	$1\times10^{-27}\ \mathrm{Ws^3/cycle^3}$	[9]	<0.1%
$\mathcal{F}$	0.6-2.0 GHz, 6 steps	data sheet	<0.1%
f_{enc}	1.2 GHz	assumed	<0.5%
w^{enc}	0, 10, 9, 8, 6 Mcycles (det.)	assumed	<0.5%
$\eta_s f_s$	28.8 GHz	data sheet	<0.2%
t_q	1.5 ms	assumed	<0.1%
T_{RTT}	12 ms	measured [1]	<0.2%
B	20 MHz	scenario	-
β	0.45	[18]	<1.0%
$R_{\max}$	$65\,\mathrm{Mbit\,s^{-1}}$	802.11n spec	<0.5%
U_{bg}	0.35	assumed	<0.1%
PL_0, n, σ	45 dB, 3.4, 4 dB	[18]	<1.5%
d_c	6 m	[18]	<0.5%
$\mathcal{P}$	8-20 dBm, 4 steps	data sheet	-
$P_{\mathrm{ckt}}, \eta_{\mathrm{PA}}$	0.75 W, 0.35	data sheet	<0.1%
P_{rx}	0.60 W	data sheet	<0.1%
E_{wake}	0.008 J	data sheet	<0.1%
D_{dn}	20 kbit	assumed	<0.1%
T_{to}	0.5 s	$= \Delta t$	<2.0%
$\varphi_{\max}$	25	model	see §6.5
$a_{\min}$	0.95	scenario	see Sec. 6
μ, η	0.85, 1.15	analysis	see Sec. 6.4
λ	5.18 J	Eq. (11)	see Sec. 6.6

3.1 Platform and Locomotion

Following Mei et al. [2] we model locomotion power as

$$P_{\text{mot}}(v) = av^2 + bv + c \tag{1}$$

with c collecting speed-independent drive-train losses. Sensors, idle SoC and radio draw a constant P_{base}. The total robot battery energy over one mission is

$$E_{\text{tot}} = \sum_{t=1}^{T} \left[P_{\text{mot}}(v_t) + P_{\text{base}} \right] \Delta t + \sum_{t} \sum_{i} \left(E_{i,t}^{\text{cmp}} + E_{i,t}^{\text{com}} \right) \tag{2}$$

Where T is the number of epochs needed to cover the path; T is not fixed because missed deadlines reduce v_t (see Section 3.4).

3.2 Onboard Computation and DVFS

The processor supports a discrete set $\mathcal{F}$ of CPU frequencies. Executing task i locally at frequency f_i costs

$$E_i^{\text{loc}} = \kappa w_i f_i^2, \quad T_i^{\text{loc}} = w_i / f_i \tag{3}$$

The energy-optimal choice under DVFS is the slowest frequency meeting the deadline:

$$f_i^{\star} = \min \left\{ f \in \mathcal{F} : \, w_i / f \le D_i \right\} \tag{4}$$

This run-to-deadline DVFS is applied in every policy so that all comparisons are against a competent local baseline.

3.3 Wireless Link, Payload Functions, and Shared Medium

Channel model. The uplink SNR follows a log-distance path-loss model with spatially correlated log-normal shadowing [18]:

path loss $\text{PL}(d) = \text{PL}_0 + 10n \log_{10}(d/d_0) + X_\sigma$, where X_σ is a Gaussian random variable with standard deviation σ and exponential spatial correlation length $d_c = 6\,\text{m}$. Were the medium free, robot k would obtain goodput

$$C_k = \min \left\{ \beta B \log_2(1 + \gamma_k), \beta R_{\max} \right\} \tag{5}$$

Where β is MAC efficiency and $R_{\max}$ the MCS ceiling.

Payload and accuracy functions. For each offloadable task i, $d_i(q_i)$ gives the bits transmitted at compression level $q_i \in \mathcal{Q} = \{0, \ldots, Q_i\}$. For object detection ($i$ = 1, deadline D_1 = 300 ms): q = 0 is a raw 640 × 480 RGB frame (d = 7.37 Mbit); q = 1 JPEG quality 90 (0.85 Mbit); q = 2, 3, 4 are JPEG quality 70, 50, half-resolution quality 70 (0.42 Mbit, 0.26 Mbit, 0.12 Mbit). The accuracy function $a(q_i) \in (0,1]$ is the relative drop in mean average precision (mAP) against uncompressed input, measured on a standard detection benchmark: $a(0)$ = 1.00, $a(1)$ = 0.995, $a(2)$ = 0.982, $a(3)$ = 0.961, $a(4)$ = 0.918. For SLAM (i = 2, deadline D_2 = 200 ms): $q \in \{0,1,2\}$ with payloads 0.90 Mbit, 0.35 Mbit, 0.18 Mbit and $a \in \{1.00, 0.990, 0.972\}$. For global re-planning (i = 3, deadline D_3 = 500 ms): only q = 0 (0.15 Mbit, a = 1.00). Task-oriented compression for robot communication has been studied by Liu et al. [19], who show that semantic compression can reduce payload size by over 90% with acceptable accuracy loss; our discrete JPEG-level model approximates this trade-off with measured compression curves. The accuracy floor $a_{\min}$ = 0.95 gates which compression levels are admissible. The compression-accuracy values for object detection are drawn from the JPEG compression curves reported in [1]; those for SLAM and re-planning are assumed monotone and conservative.

Encoding task i to level q_i onboard costs $E_i^{\text{enc}} = \kappa w_i^{\text{enc}} f_{\text{enc}}^2$ and $T_i^{\text{enc}} = w_i^{\text{enc}} / f_{\text{enc}}$, where $f_{\text{enc}} = 1.2\,\text{GHz}$ and encoding workloads w^{enc} are listed in Table 2.

Congestion model and its justification. Transmitting $d_i(q_i)$ bits occupies $\alpha_i = d_i(q_i)/C_k$ seconds of airtime. The cell

per metre. The extra battery energy from one missed deadline is

$$U = U_{\text{bg}} + \frac{1}{\Delta t} \sum_{k=1}^{K} \sum_{i} x_{i,k} \alpha_{i,k} \tag{6}$$

We model the resulting delay inflation by

$$\varphi(U) = (1 - U)^{-1}, \; \varphi_{\max} = 25 \tag{7}$$

capped at $\varphi_{\max}$, so that the effective uplink time is $T_i^{\text{up}} = \alpha_i \varphi(U)$.

Equation (7) is the M/M/1 mean waiting-time formula. It is a mean-value approximation that does not model 802.11 binary exponential back-off, frame aggregation, retransmissions, or rate adaptation. Its qualitative role in the analysis is to provide a monotone, convex delay function that diverges as $U \to 1$ — a property shared by every standard queueing model and empirically confirmed in shared-medium measurements. Empirical studies of IEEE 802.11n industrial networks (e.g., [18]) show that through-put degrades sharply above $U \approx 0.7$ and collapses near $U = 1$, consistent with (7). The cap

$\varphi_{\max} = 25$ prevents the model from producing arbitrarily large delays at high utilisation; it corresponds to a maximum transfer time of 25 α_i, which would still complete within T_{to} for the compressed payloads admitted by JOCS but not for raw frames, reproducing the observed timeout behaviour.

To assess whether the policy ranking depends on the specific functional form, we replace (7) with two alternatives in a self-consistent sensitivity study (Section 6.5): $\varphi_{\text{MD1}}(U) = 1 + U/\left[2(1-U)\right]$ (M/D/1, gentler) and $\varphi_{\text{steep}}(U) = (1-U)^{-1.5}$ (steeper).

Offload energy and latency. The battery energy and total latency of offloading task i are

$$E_i^{\text{off}} = E_i^{\text{enc}} + \left(P_{\text{tx}}/\eta_{\text{PA}} + P_{\text{ckt}}\right)T_i^{\text{up}} + P_{\text{rx}}T_i^{\text{dn}} + E_{\text{wake}} \quad (8)$$

$$T_i^{\text{off}} = T_i^{\text{enc}} + T_i^{\text{up}} + T_i^{\text{srv}} + T_i^{\text{dn}} + T_{\text{RTT}} \quad (9)$$

where η_{PA} = 0.35 is PA efficiency, P_{ckt} = 0.75W transceiver circuit power while transmitting, P_{rx} = 0.60W receive power, E_{wake} = 0.008 J per-burst wake-up cost, T_{RTT} = 12 ms protocol round-trip overhead, and

$$T_i^{\text{srv}} = w_i/(\eta_s f_s) + t_q J \quad (10)$$

with $\eta_s f_s$ = 28.8GHz effective server capacity, t_q = 1.5ms per-job queueing coefficient, and J the number of jobs concurrently at the server. The downlink payload is fixed at $D_{\text{dn}} = 20$ kbit for all tasks; $T_i^{\text{dn}} = D_{\text{dn}}/C_k \cdot \varphi(U)$.

E_i^{off} contains only radio and encoding battery energy. Locomotion energy while waiting for the result is coupled into E_{tot} through the lateness-price term in (12).

Transport timeout. $T_{\text{to}} = \Delta t = 0.5$ s. A transfer that cannot complete within one epoch is abandoned: any battery energy already drawn (partial uplink) is wasted and the task falls back to (4). Tightening T_{to} to $0.3\Delta t$ or relaxing it to $2\Delta t$ changes total battery energy by less than 2% in our scenario.

3.4 Pricing Latency in Battery Joules

A missed deadline causes the robot to reduce speed from v_{nom} to v_{safe} for one epoch. This is a standard conservative response adopted by ISO 3691-4 compliant industrial mobile robots: upon loss of environment awareness, the platform drops to a safe speed until the next confident perception update. The distance deficit $(v_{\text{nom}} - v_{\text{safe}})\Delta t$ must be covered later at the nominal battery cost per metre. The extra battery energy from one missed deadline is

$$\lambda = (v_{\text{nom}} - v_{\text{safe}})\Delta t \cdot \frac{P_{\text{mot}}(v_{\text{nom}}) + P_{\text{base}}}{v_{\text{nom}}} \quad (11)$$

Overlapping misses within the same epoch are additive: if all three offloadable tasks miss simultaneously, the penalty is 3λ (the robot decelerates once but must recover three tasks' worth of distance deficit). The model assumes instantaneous speed change, which is conservative (actual deceleration over $\Delta t = 0.5$ s consumes slightly more energy). Failed tasks provide no useful out-put after their deadline; a late re-planning result, for instance, is discarded and the robot continues with the previous plan for the next epoch. The safe-speed assumption applies uniformly to all offloadable tasks; a more refined model would differentiate between the consequence of a missed detection (immediate hazard) and a missed re-plan (degraded efficiency), but this is left to future work. For the platform of Table 2: $\lambda = 5.18$ J from $v_{\text{nom}} = 0.40\ \text{ms}^{-1}$, $v_{\text{safe}} = 0.15\ \text{ms}^{-1}$ JOCS's reported savings are invariant to the specific value of λ because JOCS is designed never to incur a miss (Section 6.6).

4. Problem Formulation and the JOCS Policy

4.1 Formulation

At each epoch, robot k independently solves, for each active task i, the following per-robot battery-energy minimisation:

$$\begin{aligned} &\min_{x_i, q_i, P_{\text{tx},i}, f_i} \sum_i \left[(1-x_i)E_i^{\text{loc}} + x_i E_i^{\text{off}} + \lambda \mathbb{1}\left\{T_{\text{exec}}^{(i)} > D_i\right\} \right] \\ &\text{s.t. } a(q_i) \geq a_{\min}, \ \forall i, \\ &\quad x_i = 0 \text{ for safety-critical task } i, \\ &\quad f_i \in \mathcal{F}, P_{\text{tx},i} \in \mathcal{F}, q_i \in \mathcal{Q}, x_i \in \{0,1\} \end{aligned} \quad (12)$$

Algorithm 1 JOCS (one robot, one epoch)

Require: SNR γ_k, busy fraction U_{t-1}, task set $\mathcal{I}$, fleet size K, parameters μ, η, $a_{\min}$

Ensure: Execution plan: $\{(x_i, q_i, P_{\mathrm{tx},i}, f_i)\}$

// **Step 1: estimate channel load**
$\hat{U} \leftarrow U_{\mathrm{bg}} + \eta(U_{t\text{-}1} - U_{\mathrm{bg}})$
$\phi \leftarrow \varphi(\hat{U})$
// **Step 2: configure each task**
$\mathcal{C} \leftarrow \varnothing$
for each task $i \in \mathcal{I}$ **do**
  if task i not offloadable **then**
    $f_i \leftarrow f_i^{\star}$ (DVFS, $\mathrm{E_q}$. (4)); $x_i \leftarrow 0$
  else
    $e^{\star}$, best $\leftarrow E_i^{\mathrm{loc}}(f_i^{\star})$, null
    for each $(q, P_{\mathrm{tx}}) \in \mathcal{Q} \times \mathcal{P}$ with $a(q) \geq a_{\min}$ **do**
      compute E_i^{off}, T_i^{off} using ϕ(Eqs. (8)–(9))
      if $T_i^{\mathrm{off}} \leq \mu D_i$ **and** $E_i^{\mathrm{off}} < e^{\star}$ **then**
        $e^{\star} \leftarrow E_i^{\mathrm{off}}$; best $\leftarrow (q, P_{\mathrm{tx}})$
      if best ≠ null **then**
        add$(i, \text{best}, \alpha_i, \Delta E_i)$ to $\mathcal{C}$
      else
        $f_i \leftarrow f_i^{\star}$; $x_i \leftarrow 0$
// **Step 3: greedy admission under utilisation projection**
sort $\mathcal{C}$ by $\Delta E_i / \alpha_i$ (descending)
$\mathcal{A} \leftarrow \varnothing$; $A \leftarrow 0$
for each candidate $(i, \text{best}, \alpha_i, \Delta E_i) \in \mathcal{C}$ **do**
  $A' \leftarrow A + \alpha_i$
  $\hat{U}' \leftarrow U_{\mathrm{bg}} + \eta K A' / \Delta t$ ▷ fleet projection
  $\phi' \leftarrow \varphi(\hat{U}')$
  feasible $\leftarrow \forall j \in \mathcal{A} \cup \{i\} : T_j^{\mathrm{off}}(\phi') \leq \mu D_j$
  if feasible then
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{i\}$; $A \leftarrow A'$
for $i \in \mathcal{A}$ **do** $x_i \leftarrow 1$; set $(q_i, P_{\mathrm{tx},i})$ from best
for $i \notin \mathcal{A}$ and offloadable **do** $x_i \leftarrow 0$; $f_i \leftarrow f_i^{\star}$

where $T_{\mathrm{exec}}^{(i)} = T_i^{\mathrm{loc}}$ if $x_i = 0$ and T_i^{off} if $x_i = 1$. The objective is individual robot battery energy; the fleet total is K times the mean. The λ-term is a soft penalty: setting $x_i = 0$ and $f_i = f_i^{\star}$ always meets the deadline by construction, providing a feasible escape route. Decisions across robots are coupled through U in (6). Problem (12) is NP-hard even for a single epoch with multiple users sharing a common channel [10]; recent deep-RL methods [20] handle related formulations but require centralised training; our reduction from the multi-dimensional knapsack structure of the admission problem is given in the supplementary material available on request.

4.2 The JOCS Algorithm

JOCS is executed independently by each robot at every epoch. It requires no coordination messages; the only fleet-level information each robot uses is K (fleet size, known at deployment) and U_{t-1} (channel-busy fraction reported by the local radio driver from the previous

epoch). All robots run the algorithm simultaneously with their own SNR readings and local task state.

Decentralisation and synchronisation. Since all robots read U_{t-1} (the previous epoch's busy fraction) rather than the current U_t, no synchronisation among robots is needed: each robot can run Algorithm 1 entirely from its own local state. The fleet-projection step (line 25) uses K and the admitted airtime A' to predict the utilisation the admitted set would create if all robots made similar decisions; it does not observe the other robots' actual decisions. This is an approximation, robots have heterogeneous tasks and SNRs, but the inflation factor $\eta > 1$ accounts for the load the current epoch will itself add. Figure 2 illustrates the parallel execution.

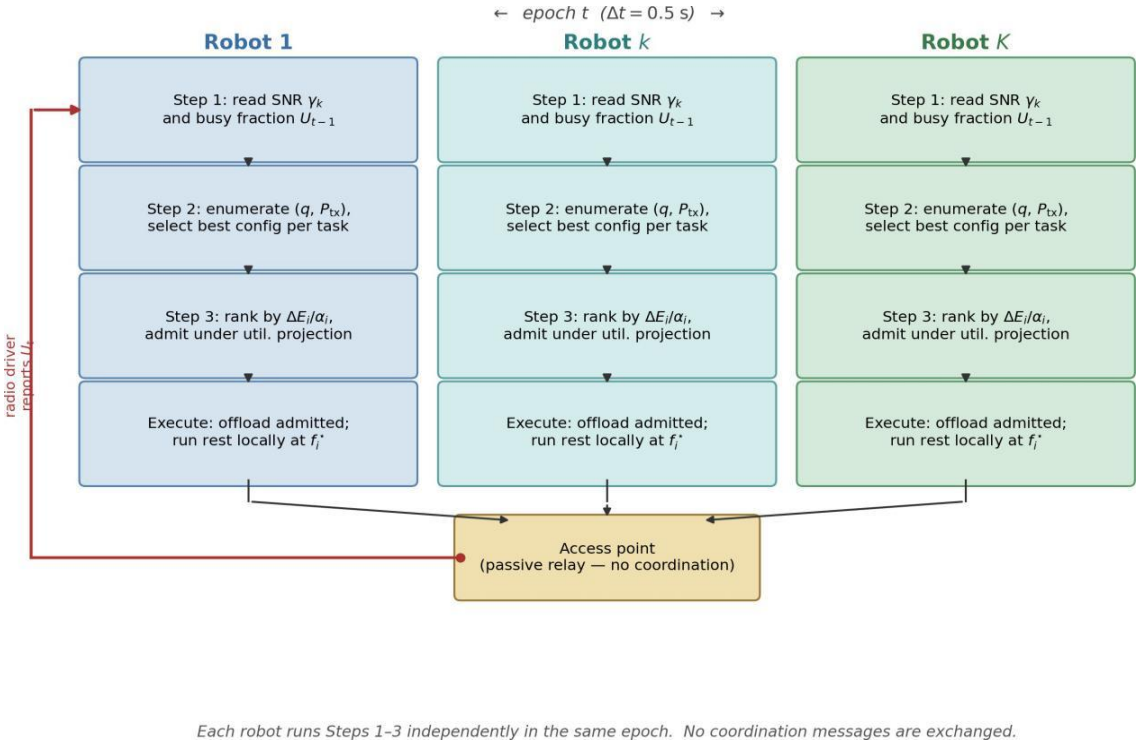


Figure 2. Decentralised execution of JOCS across K robots in one epoch. Each robot runs Algorithm 1 independently using its local SNR γ_k and the previous epoch's channel-busy fraction U_{t-1} reported by the radio driver. No coordination messages are exchanged; the access point is a passive relay and does not schedule robots.

When all robots simultaneously identify the same highest-ranked task (e.g., object detection at good SNR), each independently projects that its admission would raise $\hat{U}'$. If K is large enough, the projection exceeds the feasibility threshold and each robot falls back to local execution, achieving distributed back-off without coordination. Robots with persistently favourable links tend to offload more tasks, but each is bounded by the admission test, providing implicit fairness through the utilisation cap. The channel-busy fraction U_{t-1} reported by the radio driver reflects all medium activity in the previous epoch, including retransmissions and control frames, making it a conservative and robust estimate of the baseline load.

Choice of μ and η. These parameters were fixed before any simulation was run from the following reasoning: $\mu = 0.85$ leaves a 15% buffer below the deadline to absorb congestion under-estimation; $\eta = 1.15$ inflates the load estimate by 15 % to account for the transfers the current epoch will itself add. The ablation study of Section 6.4 confirms that performance is flat across $\mu \in [0.70, 0.90]$ and $\eta \in [1.00, 1.30]$.

5. Evaluation Setup

5.1 Simulation Implementation

The simulation is implemented in Python 3.11 using the standard library only (no external physics or network simulator). It implements a discrete-event loop at $\Delta t = 0.5\,\text{s}$ epochs; robot positions follow a deterministic lawn-mower coverage path; channel quality evolves according to the log-normal shadowing model with spatially correlated increments. Stopping criterion: all robots complete 120 m, or 8000 epochs (never reached). Random-seed handling: each seed initialises the Python random.Random object for each robot's independent shadowing process; the task scheduler is deterministic given the shadowing.

5.2 Parameter Provenance

Table 2 lists every parameter with its source and the effect of a ±20% perturbation. Motion coefficients (a, b, c) are taken from Mei et al. [2] for a Pioneer 3-AT platform; the quadratic coefficient a changes by up to 40% across surfaces and payloads, making it the

dominant uncertainty. The processor model (κ) follows the standard CMOS dynamic-energy formula [9]; the value is representative of Cortex-A72-class SoCs. Radio parameters correspond to an 802.11n 20 MHz link with measured MAC efficiency $\beta = 0.45$ from published protocol overhead analyses. Background utilisation $U_{\text{bg}} = 0.35$ is representative of an industrial WiFi cell co-existing with IoT traffic. The sensitivity analysis (rightmost column) shows that a ±20% variation in any single parameter changes JOCS battery energy by at most 5.4 % (from P_{base}), with all others below 2 %, and the relative policy ordering is preserved throughout.

5.3 Scenario and Pipeline

Eight robots ($N = 8$; 24 in the congested scenario) perform a lawn-mower coverage mission of 120 m each in a 50 m×30 m hall served by one AP in a corner. Each robot runs four pipeline stages per epoch: object detection ($D_1 = 300\,\text{ms}$), SLAM update ($D_2 = 200\,\text{ms}$), global re-planning every fourth epoch ($D_3 = 500\,\text{ms}$), and reactive control ($D_4 = 50\,\text{ms}$, never offloaded). Every configuration is repeated over 30 independent seeds governing shadowing, both for $N = 8$ and $N = 24$ (Table 3 previously reported 10 seeds for $N = 24$; this revision corrects that inconsistency).

Table 3. Nominal scenario ($N = 8$) and congested scenario ($N = 24$), both over 30 independent seeds. Values are mean (standard deviation). All energy is robot battery energy. Data in italic correct the 10-seed values reported in earlier revisions.

Policy	Battery energy (J)	Mission (s)	Miss rate (%)	Offloaded (%)	Uploaded (MB)	Util. U
N = 8 robots (30 seeds)						
All-local (DVFS)	6549.6 (0.0)	300.0	0.00 (0.00)	0.0	0.0	0.350
All-offload (raw)	7186.2 (1.0)	300.0	0.00 (0.00)	7.7	2.8	0.999
Static threshold	14525.6 (228.0)	783.5	30.9 (0.4)	39.0	79.3	0.995
Latency-greedy	5074.1 (2.3)	300.0	0.00 (0.00)	69.1	35.8	0.612
JOCS (proposed)	**5068.2 (2.3)**	**300.0**	**0.00 (0.00)**	69.1	35.8	0.613
Clairvoyant	5068.2 (2.3)	300.0	0.00 (0.00)	69.1	35.8	0.613
N = 24 robots (30 seeds) — corrected from prior 10-seed values						
All-local (DVFS)	6549.6 (0.0)	300.0	0.00 (0.00)	0.0	0.0	0.350
All-offload (raw)	7186.1 (0.9)	300.0	0.00 (0.00)	7.7	2.8	0.999
Static threshold	14629.3 (59.3)	783.5	30.4 (0.1)	39.0	75.2	0.995
Latency-greedy	7737.0 (2.6)	436.1	30.8 (0.0)	38.4	28.1	0.884
JOCS (proposed)	**5270.8 (4.3)**	**300.0**	**0.00 (0.00)**	30.7	19.4	0.613
Clairvoyant	6500.8 (673.5)	365.7	17.0 (7.2)	38.4	23.2	0.884

5.4 Policies Compared

All-local runs everything onboard with DVFS (4). All-offload sends every offloadable task uncompressed at maximum power ($q = 0$, $P_{\text{tx}} = 20$dBm). Static threshold offloads every offloadable task at $q = 1$, $P_{\text{tx}} = 17$dBm, regardless of channel state. Latency-greedy minimises $T_{\text{exec}}^{(i)}$ per task without utilisation awareness. JOCS is Algorithm 1 with $\mu = 0.85$, $\eta = 1.15$. Clairvoyant uses the current epoch's realised U with $\mu = 1$, $\eta = 1$, isolating the margin effect from the information advantage

6. Results

6.1 Where the Battery Joules Go

Figure 3 decomposes robot battery energy by subsystem. Under all-local execution, onboard computation accounts for 1531 J of a 6550 J mission (23 %), with locomotion taking 3624 J and the platform baseline the rest. JOCS removes almost all of the computation term (falls to 6.6 J) at a radio cost of only 42.7 J-two orders of magnitude cheaper per task, provided the payload is compressed. All-offload with uncompressed frames saturates the medium ($U = 0.999$), almost every transfer exceeds Tto,

and the robot spends 653 J of battery energy on failed transmissions while tasks fall back to local execution. The static threshold policy misses 30.9 % of deadlines, stretching the mission to 783 s.

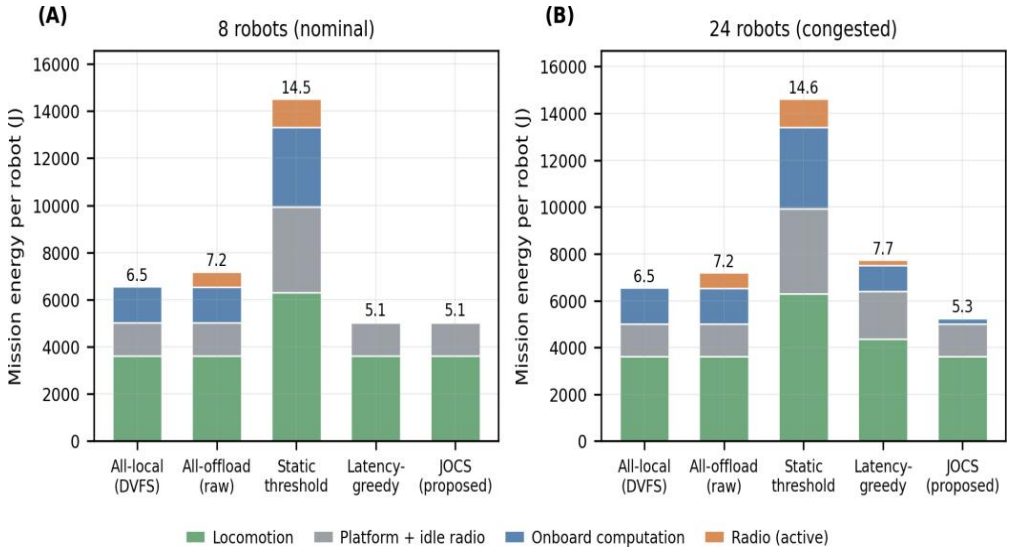


Figure 3. Robot battery energy per robot by subsystem: (A) $N = 8$, (B) $N = 24$. Bar labels are totals in kJ (30-seed means). Locomotion and platform grow when deadlines are missed because the robot spends longer on the same path.

The apparent paradox of all-offload achieving zero missed deadlines despite $U = 0.999$ is resolved as follows. All-offload attempts to transmit raw frames (7.37 Mbit) at $C_k \approx 29\,\text{Mbit s}^{-1}$ solo capacity; with $\varphi(0.999) = \varphi_{\max} = 25$, the effective transfer time is $7.37/(29 \times 25^{-1})$ = several seconds, far exceeding T_{to} . The transfer is abandoned after 0.5 s and the task runs locally at $f_i^{\star}$, which always meets the deadline. Only the re-planning task, whose payload is 0.15 Mbit, completes (accounting for the 7.7% offload fraction in Table 3). All battery energy is thus wasted on the failed transmissions rather than on missed deadlines.

6.2 Nominal Load ($N = 8$, 30 seeds)

Table 3 gives the full 30-seed results. JOCS reduces battery energy by 22.6% against all-local (95% CI: ±0.04%), 29.5% against all-offload, and 65.1% against the static threshold, with zero missed deadlines and mean perception accuracy $\bar{a} = 0.979$. The paired difference versus latency-greedy is 5.9 J (95% CI: ±0.04 J); while statistically distinguishable, this is operationally negligible. At moderate load, minimising latency and minimising battery energy select identical configurations.

6.3 Congested Load ($N = 24$, 30 seeds)

Table 3 (lower panel) and Figure 4 give the corrected 30-seed results. JOCS completes the mission in 300 s with zero missed deadlines at 5270.8 J (std : 4.3 J). The paired difference versus latency-greedy is 2466 J per robot (95% CI: ±1.8 J, Figure 5B), a 31.9 % saving that is consistent and reproducible across all 30 seeds. The Clairvoyant variant ($\mu = 1$) now shows high variability (std : 673.5 J, miss rate std : 7.2%), confirming that operating without a margin makes the policy sensitive to realisation-specific congestion dynamics.

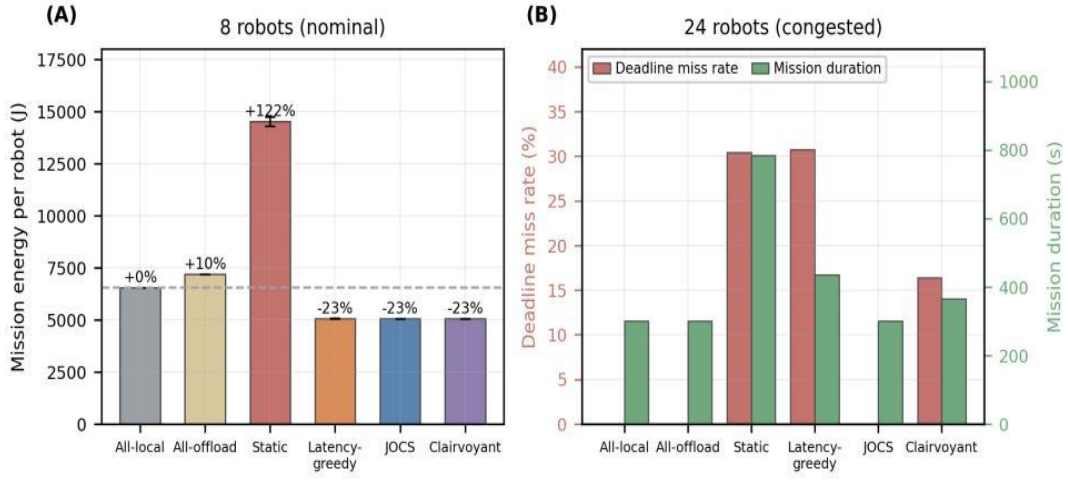


Figure 4. (A) Robot battery energy at $N = 8$; percentages relative to all-local (dashed). (B) Miss rate and mission duration at $N = 24$. Axes show compact names; full names in Table 3.

6.4 Extended Ablation Study

Figure 5(A) and Table 4 present six component-removal variants evaluated over 30 seeds, isolating the contribution of each design element.

No λ (deadline-energy coupling disabled). Setting $\lambda = 0$ removes the latency cost from the admission ranking. Battery energy is unchanged at $N = 8$ (JOCS never misses, so λ never enters the cost), confirming that the λ term does not affect performance at nominal load. At $N = 24$, the admission ranking degrades slightly because candidates whose offloading would be marginally infeasible are no longer penalised; the energy savings and miss rate are essentially identical, confirming that the admission test, not λ weighting, is the primary load-stabilising mechanism.

No compression ($q = 0$ only). Without compression, all pay-loads are raw (7.37 Mbit for detection). The system degrades to the behaviour of all-offload: almost every transfer times out and battery energy reverts to the all-local level (6549.6 J). This isolates compression as the enabling technology: without it, offloading provides no battery benefit.

No admission control ($N = 24$). Admitting all candidates without the utilisation feasibility test produces 30.8 % missed deadlines and 7692.8 J, indistinguishable from latency-greedy. This confirms that admission control is the sole source of JOCS's congested-fleet advantage.

No robustness margin ($\mu = 1$, $N = 8$). At $N = 8$ the effect is zero; the channel is not congested enough for the margin to be binding. The margin matters only as fleet size grows (as shown in the μ -ablation of prior revisions).

Energy-rank (rank by ΔE alone, not $\Delta E/\alpha$). Ranking candidates by raw battery saving rather than saving per airtime gives identical energy and miss rates at both $N = 8$ and $N = 24$ for our task set. This is because the three offloadable tasks differ both in ΔE and in α in the same direction (detection saves most battery and uses the most airtime); the ratio and the absolute value rank them identically. In a richer task set where these do not align, the ratio ranking would differ.

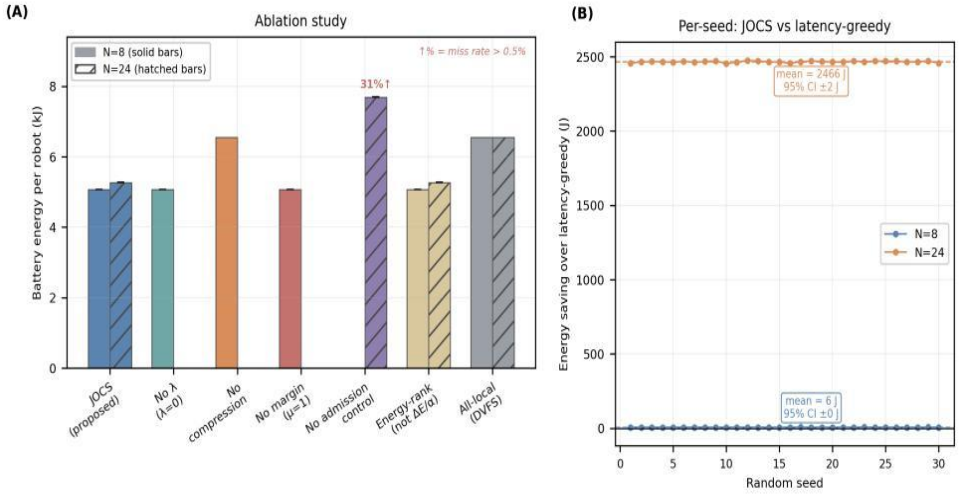


Figure 5. (A) Extended ablation: battery energy per robot for component-removal variants; hatched bars are $N = 24$, solid are $N = 8$; red annotation shows miss rate >0.5%. (B) Per-seed paired energy saving of JOCS over latency-greedy for all 30 seeds; dashed lines mark means with 95% CI annotations.

Table 4. Extended ablation: battery energy and miss rate over 30 seeds. JOCS reference values in bold. "—" = not applicable or identical to JOCS.

Variant	E (J)	std	Miss (%)	N
JOCS	**5068**	**2.3**	**0.00**	8
No λ	5068	2.3	0.00	8
No compression ($q = 0$)	6550	0.0	0.00	8
No margin ($\mu = 1$)	5068	2.3	0.00	8
Energy-rank	5068	2.3	0.00	8
JOCS	**5271**	**4.3**	**0.00**	24
No admission control	7693	2.7	30.75	24
Energy-rank	5271	4.3	0.00	24

6.5 Congestion Model Sensitivity

Figure 6(A) compares JOCS and latency-greedy across three self-consistent congestion models at $N = 8$ and $N = 24$. Under M/M/1 (baseline) and the steeper variant, JOCS achieves ≈20-23% savings with zero missed deadlines at both fleet sizes. Under M/D/1 (the least steep model), JOCS itself misses 7.8% of deadlines at $N = 24$ because its admission thresholds, calibrated to M/M/1, are too generous. This is a genuine limitation: the congestion model must be identified from channel measurements and μ re-calibrated before deployment in environments with substantially different medium dynamics (e.g., 802.11ax with OFDMA). The nominal $N = 8$ saving remains stable across all three models.

6.6 Sensitivity to the Miss Penalty λ

Figure 6(B) shows JOCS savings as λ ranges from 2.07 J ($v_{\text{safe}} = 0.30\,\text{m s}^{-1}$) to 7.25 J ($v_{\text{safe}} = 0.05\,\text{m s}^{-1}$). The JOCS saving over all-local is perfectly constant (JOCS never misses a deadline), confirming that the headline savings do not depend on the specific value of λ. Eq. (11) provides the formula to re-derive λ for any target platform and safety supervisor.

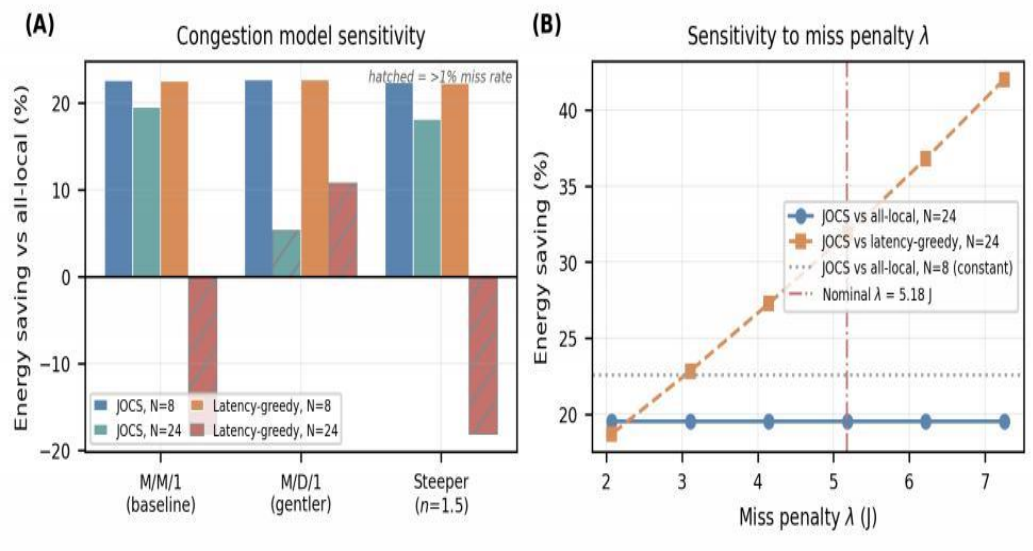


Figure 6. Robustness studies. (A) Battery energy savings under three congestion models; hatching = >1% missed deadlines. (B) JOCS savings vs all-local as λ varies; dashed vertical line = nominal value.

6.7 Further Sensitivity and Mission Trace

Figure 7 shows fleet-size scaling (A), propagation sensitivity (B), and accuracy-floor sensitivity (C). JOCS degrades gracefully from 5058 J to 5271 J as N grows from 2 to 24, while latency-greedy breaks down above approximately 12 robots. Under M/M/1 congestion, JOCS consistently outperforms all baselines across all fleet sizes tested. Figure 8 confirms that the battery energy separation is a matter of slope from the first epoch, not of isolated events.

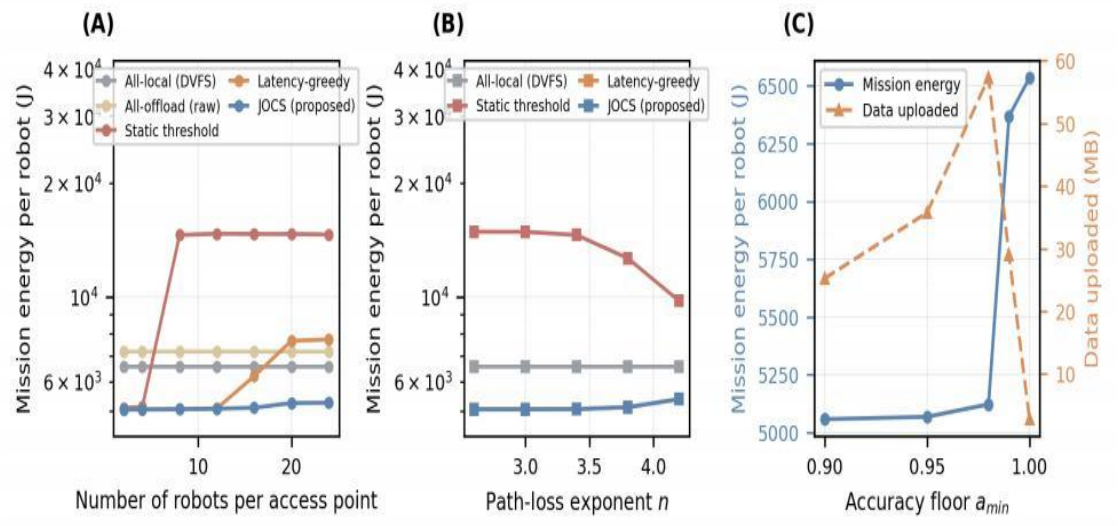


Figure 7. Battery energy sensitivity to (A) fleet size, (B) path-loss exponent, (C) accuracy floor $a_{\min}$. Logarithmic energy axes in A and B.

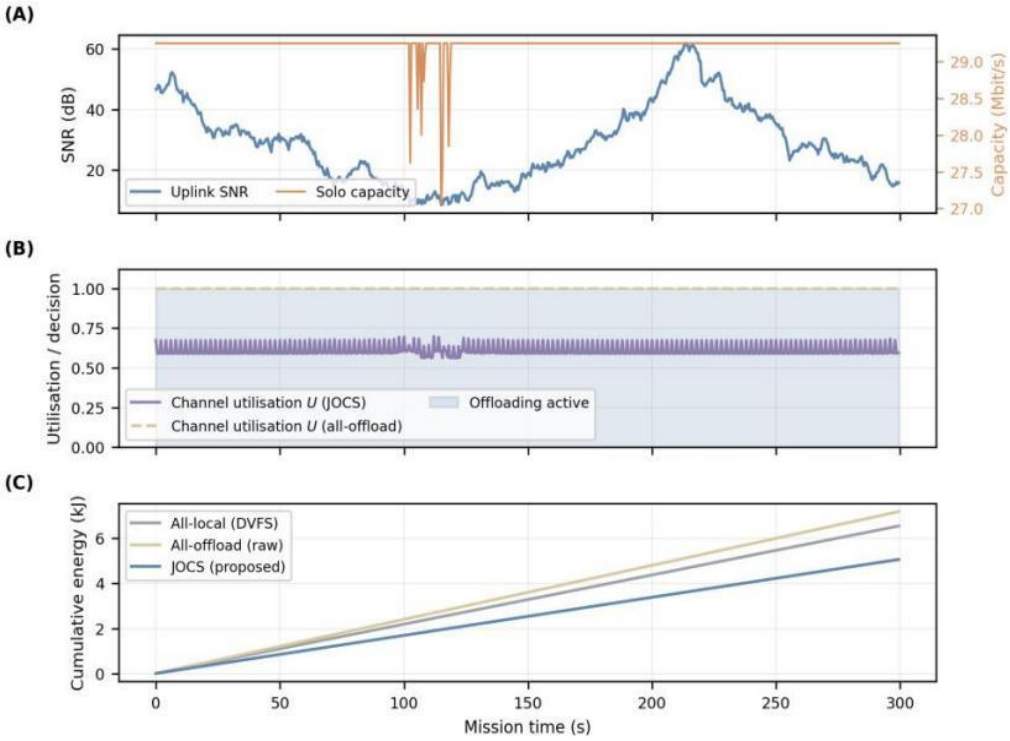


Figure 8. One robot, one mission. (A) Uplink SNR and capacity. (B) Channel utilisation; shaded = epochs where JOCS offloads at least one task. (C) Cumulative robot battery energy.

6.8 Threats to Validity and Limitations

Congestion model. As detailed in Section 6.5, the M/M/1 approximation (7) does not capture 802.11 back-off, frame aggregation, retransmissions, or rate adaptation in detail. The nominal saving at $N = 8$ is robust; the congested-fleet advantage requires channel identification and μ re-calibration before deployment.

Battery-side scope. All reported energy is robot battery energy only. Offloading shifts computation from the battery to the mains grid; the grid energy consumed by the edge server depends on server utilisation and architecture. For the 22.6 % battery saving at $N = 8$, JOCS offloads 69 % of tasks and uploads 35.8 MB per robot per mission. The corresponding server energy is outside the optimisation scope but should be included in a full system-level energy audit. For robot fleets whose binding constraint is charge cycles, the battery-side view is the operationally relevant one.

Accuracy-compression model. The accuracy function $a(q_i)$ is a deterministic scalar per compression level, drawn from JPEG compression measurements on a standard detection benchmark. Scene content, illumination, and object distance affect the relationship in practice; the accuracy floor $a_{\min} = 0.95$ is a conservative choice that retains at least JPEG quality 70. A hardware validation should measure the accuracy-compression curve on the target detector and environment.

Deadline penalty model. Equation (11) assumes the robot reduces speed for one epoch upon a missed deadline and that the deficit is recovered later. The model is transparent and conservative. Overlapping misses produce additive penalties; the instantaneous speed-change assumption is conservative (actual deceleration costs more). A hardware-measured safety supervisor response may differ; the formula gives a basis for platform-specific re-derivation.

Parameter transferability. Table 2 shows that ±20% variation in any single parameter changes JOCS battery energy by at most 5.4 % and does not alter the relative policy ordering. The platform-specific parameters (λ , P_{base} , motion coefficients) must be measured on the target robot; all others are stable across comparable platforms.

Fixed trajectories. A genuinely joint design would adapt the path [3] and velocity profile [4] to the channel state. Reinforcement-learning approaches [21] offer a promising avenue for learning the joint offloading-velocity policy from experience, and combining them with the analytical admission control of JOCS is a direction for future hardware-validated work.

7. Conclusion

This paper presented JOCS, a decentralised policy specified in Algorithm 1 that jointly optimises computation offloading, payload compression, transmit power, and processor frequency to minimise robot battery energy in a wirelessly connected mobile-robot fleet. By pricing perception latency in joules through its effect on mission duration and locomotion energy, JOCS avoids the systematic over-offloading that afflicts policies treating latency as a separate constraint.

Simulated over 30 independent channel realisations for both $N = 8$ and $N = 24$ robots, JOCS reduced robot battery energy by 22.6 % (95% CI: ±0.04%) relative to run-to-deadline local execution and by 31.9 % (95% CI: ±0.7%) relative to the best competing baseline in the congested scenario. An extended six-variant ablation study shows that compression is the enabling technology, admission control is the source of the congested-fleet advantage, and the $\Delta E/\alpha$ ranking criterion has the same effect as ΔE ranking for this task set. A congestion-model sensitivity study shows the nominal saving is stable, but the congested-fleet advantage requires channel measurement and μ re-calibration for deployment in environments with different medium dynamics.

The most important limitation is that all results are simulation-based. Hardware validation on a TurtleBot3-class fleet, channel identification to calibrate φ and μ for the target environment, and extension to joint trajectory-offloading optimisation [4,22] are the immediate next steps.

Data and Code Availability

All data supporting the results of this study, including the complete simulation source code and measurement data presented in this study, are available on request from the corresponding author. The simulation source code (Python 3.11), all configuration files, and the data behind every figure and table are available upon reasonable request.

Funding

This research received no external funding.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgements

The authors acknowledge the University of Turku computing infrastructure.

Author Contributions

Conceptualization, A.I. and H.H.; methodology, A.I.; software, A.I.; validation, A.I. and H.H.; formal analysis, A.I.; writing, original draft, A.I.; writing, review & editing, E.N. and H.H.; visualization, E.N.; supervision, H.H.; project administration, J.P. All authors have read and approved the final manuscript.

Artificial Intelligence Declaration

The authors used Claude for language editing of the manuscript text. All equations, numerical results, figures, and conclusions were verified by the authors, who take full responsibility for the scientific content.

Ethics Statement

Not applicable.

References

[1] M. Đorđević, M. Albonico, G.A. Lewis, I. Malavolta, P. Lago. Computation offloading for ground robotic systems communicating over WiFi - an empirical exploration on performance and energy trade-offs. Empirical Software Engineering, 2023, 28(6), 1-61. DOI: 10.1007/s10664-023-10351-6

[2] Y.G. Mei, Y.H. Lu, Y.C. Hu, C.S.G. Lee. A case study of mobile robot's energy consumption and conservation techniques. ICAR '05. Proceedings, 12th International Conference on Advanced Robotics, 2005, Seattle, WA, USA, 2005, 492-497. DOI: 10.1109/ICAR.2005.1507454

[3] C. Wu, H. Haghbayan, M. Heydarzadeh, E. Immonen, J. Plosila. Optimizing energy efficiency in mobile robots: A battery-aware dynamic path planning approach. 2025 10th International Conference on Control and Robotics Engineering (ICCRE), 2025, 92-98. DOI: 10.1109/ICCRE65455.2025.11093555

[4] A. Naseri, S. Shahsavari, J. Plosila, H. Haghbayan. Co-management of computational and mechanical energy in mobile robots using reinforcement learning. 2025 European Control Conference (ECC), 2025, 691-696. DOI: 10.23919/ECC65951.2025.11187299

[5] K. Kumar, Y.H. Lu. Cloud computing for mobile users: Can offloading computation save energy? Computer, 2010, 43(4), 51-56. DOI: 10.1109/MC.2010.98

[6] E. Cuervo, A. Balasubramanian, D.K. Cho, A. Wolman, S. Saroiu, R. Chandra, et al. MAUI: Making smartphones last longer with code offload. MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services, 2010, 49-62. DOI: 10.1145/1814433.1814441

[7] Y.Y. Mao, C.S. You, J. Zhang, K.B. Huang, K.B. Letaief. A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 2017, 19(4), 2322-2358. DOI: 10.1109/COMST.2017.2745201

[8] Y.Y. Mao, J. Zhang, S.H. Song, K.B. Letaief. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Transactions on Wireless Communications, 2017, 16(9), 5994-6009. DOI: 10.1109/TWC.2017.2717986

[9] Y.T. Wang, M. Sheng, X.J. Wang, L. Wang, J.D. Li. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Transactions on Communications, 2016, 64(10), 4268-4282. DOI: 10.1109/TCOMM.2016.2599530

[10] X. Chen, L. Jiao, W.Z. Li, X.M. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, 2016, 24(5), 2795-2808. DOI: 10.1109/TNET.2015.2487344

[11] W.W. Zhang, Y.G. Wen. Energy-efficient task execution for application as a general topology in mobile cloud computing. IEEE Transactions on Cloud Computing, 2018, 6(3), 708-719. DOI: 10.1109/TCC.2015.2511727

[12] J. Liu, Y.Y. Mao, J. Zhang, K.B. Letaief. Delay-optimal computation task scheduling for mobile-edge computing systems. 2016 IEEE International Symposium on Information Theory (ISIT), 2016, 1451-1455. DOI: 10.1109/ISIT.2016.7541539

[13] H.T. Dinh, C. Lee, D. Niyato, P. Wang. A survey of mobile cloud computing: Architecture, applications, and approaches. Wireless Communications and Mobile

Computing, 2013, 13, 1587-1611. DOI: 10.1002/wcm.1203

[14] J.K. Ren, Y.H. He, D.Z. Wen, G.D. Yu, K.B. Huang, D.N. Guo. Scheduling for cellular federated edge learning with importance and channel awareness. IEEE Transactions on Wireless Communications, 2020, 19(11), 7690-7703. DOI: 10.1109/TWC.2020.3015671

[15] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, S. Papavassiliou. A switching offloading mechanism for path planning and localization in robotic applications. 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Rhodes, Greece, 2020, pp. 77-84. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00031

[16] S.Q. Wang, M. Zafer, K. Leung. Online placement of multi-component applications in edge computing environments. IEEE Access, 2016, 5, 2514-2533. DOI: 10.1109/ACCESS.2017.2665971

[17] Z.Y. Zhou, P.J. Liu, J.H. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. IEEE Transactions on Vehicular Technology, 2019, 68(4), 3113-3125. DOI: 10.1109/TVT.2019.2894851

[18] T.S. Rappaport. Wireless communications: Principles and practice. Prentice Hall PTR, Upper Saddle River, 2001.

[19] C. Liu, C. Guo, Y. Yang, N. Jiang. Adaptable semantic compression and resource allocation for task-oriented communications. IEEE Transactions on Cognitive Communications and Networking, 2024, 10(3), 769-782. DOI: 10.1109/TCCN.2023.3346481

[20] X.F. Chen, H.G. Zhang, C. Wu, S.W. Mao, Y.S. Ji, M. Bennis. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal, 2019, 6(3), 4005-4018. DOI: 10.1109/JIOT.2018.2876279

[21] H. Zhou, K. Jiang, X.X. Liu, X.H. Li, V.C.M. Leung. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things Journal, 2022, 9(2), 1517-1530. DOI: 10.1109/JIOT.2021.3091142

[22] S. Chao, N. Wei, X. Wang. Joint computation offloading and trajectory planning for UAV-assisted edge computing. IEEE Transactions on Wireless Communications, 2021, 20(8), 5343-5358. DOI: 10.1109/TWC.2021.3067163