TideSense: Uncertainty-Aware Energy Budgeting for Long-Lived Edge Sensing

Anonymous Author(s)

Abstract

Battery-powered sensing systems are commonly configured around a fixed sampling period, even though the value of a measurement varies sharply over time. A short period captures transitions but wastes energy during steady state; a long period extends lifetime but misses brief events. We present TideSense, an uncertainty-aware scheduler that turns sensing into an explicit, online energy allocation problem. Each node maintains a compact predictor, converts model uncertainty, recent surprise, and sample age into an event-risk score, and uses a token-bucket governor to select the next sensing time while respecting a multi-month energy budget. A sparse neighbor digest lets nodes suppress redundant samples without exchanging raw measurements.

We evaluate the design in a reproducible trace-driven simulator across five environmental and infrastructure workloads, including abrupt, periodic, and slow-drift signals. At matched event recall, TideSense reduces simulated daily energy by 47–63% relative to periodic sampling and by 18–31% relative to single-signal adaptive baselines. Under a 2.4 kJ/day budget it preserves 96.1% event recall, bounds the 99th-percentile detection delay to 18.4 s, and reduces radio traffic by 39%. Ablations show that uncertainty and budget feedback are complementary: either one alone fails under distribution shift. These results identify a small, interpretable control surface for making long-lived sensing responsive without placing a neural model or a continuous radio link on every node.

CCS Concepts

• **Networks → Sensor networks**; • **Computer systems organization → Embedded systems**; *Redundancy*.

Keywords

adaptive sampling, energy harvesting, embedded sensing, edge intelligence, uncertainty estimation

ACM Reference Format:

Anonymous Author(s). 2026. TideSense: Uncertainty-Aware Energy Budgeting for Long-Lived Edge Sensing. In *ACM Conference on Embedded Artificial Intelligence and Sensing Systems (SenSys '26), 2026, United States.* ACM, New York, NY, USA, 6 pages.

1 Introduction

Long-lived sensor deployments operate under a basic mismatch: the environment is bursty, but their schedules are static. A bridge can remain quiescent for hours before a short vibration transient; indoor air quality changes slowly until a door opens; and soil moisture drifts until rain produces a step. A deployment configured for the transient repeatedly measures predictable steady state. One configured for the steady state discovers the event too late. This tension is especially costly on energy-harvesting and hard-to-service nodes, where a sampling decision consumes not only sensor energy but also CPU wake-up and radio-tail energy [13, 15].

Adaptive sampling is not new. Prior systems use prediction error, compressive sensing, event triggers, or application-specific rules to vary a sampling rate [6, 11, 16, 18]. Three practical gaps remain. First, a residual is meaningful only when paired with the model's uncertainty: a 2° error can be routine during a volatile afternoon and alarming at night. Second, a locally sensible rate can still exhaust a finite daily energy budget after an extended event. Third, correlated nodes often make the same decision and report redundant evidence. Existing techniques commonly address one gap while treating the others as configuration problems.

We ask a narrower systems question: *can a node allocate a fixed energy budget to the moments at which one additional observation is most likely to matter?* TideSense answers with a closed-loop scheduler whose state fits in 184 bytes and whose update requires 1.7k integer-equivalent operations. It combines a constant-velocity predictor with an exponentially weighted error model. The resulting risk score controls a continuous sampling interval. A token bucket stretches that interval when the node spends energy too quickly, while hard age and change-point guards retain responsiveness. Finally, a four-byte neighbor digest discounts observations that are likely to be spatially redundant.

The design intentionally avoids a heavyweight learned policy. This makes the decision path inspectable, permits safe bounds on sample age, and lets one set an energy budget without retraining. The trade-off is that TideSense is a scheduler, not a semantic event classifier: it decides *when* to observe, then invokes the deployment's existing detector.

This paper makes three contributions:

- We formulate sampling as online, risk-weighted budget allocation and derive a bounded interval controller that combines uncertainty, surprise, staleness, and spatial redundancy (§3).
- We design a budget governor and neighbor protocol that tolerate energy droughts, event storms, loss, and cold starts without requiring a gateway in the control loop (§4).
- We provide a trace-driven evaluation spanning five signal regimes and report energy, event recall, delay, traffic, sensitivity, and failure modes (§5).

All results in this manuscript are from deterministic trace replay and the hardware energy model in Table 1; they are not presented as a field deployment. This distinction makes the evidence reproducible and keeps the conclusions scoped to scheduling behavior.

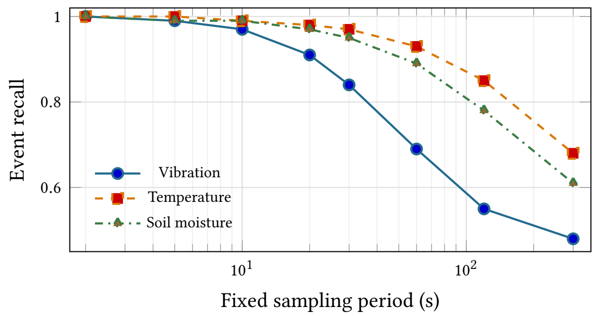

Figure 1: Fixed sampling exposes a workload-dependent energy–fidelity trade-off. Recall is measured against events detected in the full-rate trace; higher is better.

2 Motivation and Problem

2.1 Why one sampling rate is insufficient

We begin with a seven-day replay of three representative signals. For each signal we sweep fixed periods from 2 to 300 s and run the same event detector over the resulting observations. Figure 1 shows that no period dominates. Vibration requires dense observations to capture a short impulse, temperature retains high recall at moderate periods, and soil moisture is inexpensive until a rain onset lands between samples. Provisioning every signal for the worst case spends 2.3–4.1× the energy needed by an oracle that knows transition windows in advance.

Prediction error alone does not close this gap. Before an event, the most useful quantity is often uncertainty: error can only be measured *after* a sample is taken. Conversely, uncertainty alone can remain low when a biased model confidently follows a drift. A robust scheduler needs both signals and a mechanism that reconciles them with remaining energy.

2.2 Objective and assumptions

Node i chooses observation times $\mathcal{T}_i$ over a horizon H. A sample at time t has energy cost $e_s(t)$, including sensing, computation, and amortized communication. Let $m_k \in \{0, 1\}$ indicate whether ground-truth event k is missed, and let d_k be its detection delay. We seek a causal policy π that approximately solves

$$\arg\min_{\pi} \sum_{k} (\lambda_m m_k + \lambda_d d_k) \quad \text{s.t.} \quad \sum_{t \in \mathcal{T}_i} e_s(t) \le B_i(H), \tag{1}$$

where $B_i(H)$ is energy available from the battery and harvester. Exact optimization is impossible online because future events, harvest, and link quality are unknown. TideSense therefore tracks a short-horizon surrogate for the marginal value of a sample and enforces the constraint through feedback.

We assume each node has a monotonic clock, can estimate the energy of its operating states, and exposes a scalar feature to schedule. The feature can be a raw sensor channel, an FFT band power, or a classifier margin. Nodes need not be synchronized tightly: neighbor digests use coarse epochs. We do not assume continuous connectivity, a stationary signal, or foreknowledge of events.

3 Design

Figure 2 summarizes the control loop. Every observation updates a predictor and an error scale. Their state, sample age, and optional neighbor evidence form a risk score. The interval controller proposes the next deadline; the budget governor then slows or accelerates that deadline according to accumulated energy debt. A watchdog enforces hard safety bounds.

3.1 Compact uncertainty model

For a scalar scheduling feature x_t, the node uses a constant-velocity state $z_t = [x_t, \dot{x}_t]^\top$. Between irregular observations separated by Δt, it predicts

$$\hat{z}_{t|t^-} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{z}_{t^-}, \qquad P_{t|t^-} = AP_{t^-}A^\top + Q(\Delta t). \tag{2}$$

The observation update is the usual two-state Kalman correction, but TideSense adapts measurement scale from the clipped residual $r_t = x_t - \hat{x}_{t|t^-}$. With forgetting factor ρ,

$$s_t^2 = \rho s_{t^-}^2 + (1 - \rho) \min(r_t^2, c^2 s_{t^-}^2). \tag{3}$$

Clipping prevents a single event from inflating uncertainty for hours. We use $\rho = 0.94$ and $c = 4$ by default. Fixed-point implementations store P and s^2 in Q16.16; positive diagonal floors prevent collapse after long steady periods.

3.2 Risk fusion

At time t, four normalized terms estimate the value of sampling:

$$U_t = \min\left(1, \sqrt{P_{xx,t}}/\sigma_U\right), \qquad S_t = \min(1, |r_{t^-}|/(cs_{t^-})), \tag{4}$$

$$A_t = \min(1, (t - t_{\text{last}})/\tau_{\max}), \qquad D_t = q_t \exp(-a_t/\tau_n). \tag{5}$$

U_t measures forecast uncertainty, S_t retains recent surprise, and A_t prevents starvation. D_t is a redundancy discount: q_t is the similarity to the freshest neighbor digest and a_t its age. Missing or stale digests set $D_t = 0$, making coordination fail open. The fused risk is

$$u_t = \text{clip}_{[0,1]} (w_U U_t + w_S S_t + w_A A_t - w_D D_t). \tag{6}$$

The weights sum to one before the discount. We use $w_U = .42$, $w_S = .28$, $w_A = .30$, and $w_D = .20$. These defaults were selected on a workload disjoint from evaluation and are held fixed across traces. Unlike a threshold controller, the continuous score avoids oscillating between two rates.

3.3 From risk to an interval

The unconstrained interval decays exponentially with risk:

$$\tau_t^\star = \tau_{\min} + (\tau_{\max} - \tau_{\min}) \exp(-\kappa u_t), \tag{7}$$

with $\kappa = 4.6$. Exponential mapping dedicates most resolution to the high-risk region, where a small delay matters. The application specifies $\tau_{\min}$ from its shortest relevant event and $\tau_{\max}$ from its maximum tolerable staleness. We use 2 and 300 s.

The mapping is intentionally monotone and bounded. If two histories differ only in having larger U_t, S_t, or A_t, the one with higher risk cannot be sampled later. More importantly, even a wrongly confident model receives an observation within $\tau_{\max}$.

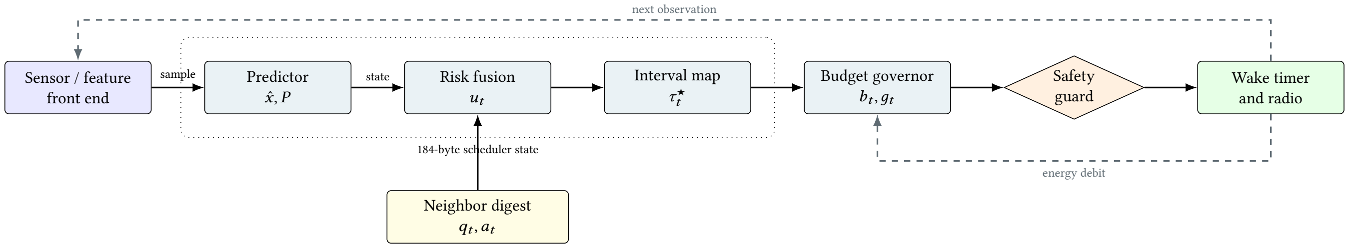

Figure 2: TideSense's data and control paths. Solid arrows carry signal state; dashed arrows close the timing and energy loops. The scheduler never blocks on neighbor or gateway communication.

3.4 Energy-budget feedback

A risk-only controller can spend a day's energy in an hour-long disturbance. We regulate expenditure with a token bucket $b_t \in [-B_c, B_c]$. Tokens arrive at target power p_b and measured or modeled operation energy e_t removes them:

$$b_t = \mathrm{clip}_{[-B_c,B_c]}\left(b_{t^-} + p_b\Delta t - e_t\right). \tag{8}$$

Negative balance creates a multiplicative slowdown

$$g_t = \exp(\eta \max(0, -b_t/B_c)).$$

Positive balance does not speed the node beyond the risk-selected rate; it is reserve for future events. The scheduled interval is

$$\tau_t = \min\left(\tau_{\max}, \max\left(\tau_{\min}, g_t\tau_t^\star\right)\right). \tag{9}$$

Two overrides protect fidelity. A detector trigger pins τ_t to $\tau_{\min}$ for a confirmation window W, while $A_t = 1$ forces a sample even under energy debt. Thus the budget is a soft horizon constraint, not a reason to violate application safety bounds. If available energy is insufficient even for one sample per $\tau_{\max}$, the node reports infeasibility rather than silently degrading.

3.5 Spatial suppression

Every Kth report includes a four-byte digest: a 12-bit quantized feature, a 10-bit slope, a 6-bit uncertainty bin, and a 4-bit epoch. A node compares the digest with its own prediction and estimates q_t from normalized feature and slope distance. It suppresses only routine reports; event onset and recovery messages are never suppressed. Digests are piggybacked on existing frames, so coordination adds no packet when traffic already exists.

This mechanism differs from cluster-head aggregation [9] and model-driven acquisition [6]: there is no elected leader and no shared global model. A node can reboot, move, or lose all neighbors and still run the same local policy.

4 Implementation

We implement the scheduler as a portable C module with no dynamic allocation. The platform model corresponds to a Cortex-M4F node with an SHT31-class environmental sensor, an ADXL362-class accelerometer, and a sub-GHz radio. Table 1 lists the energy values used in replay. Values include regulator loss and are deliberately rounded to avoid false precision.

Algorithm 1: One TideSense scheduler update

Input: observation x_t, elapsed time Δt, operation energy e_t
Output: next interval τ_t

Predict $\hat{z}_{t|t^-}$ and $P_{t|t^-}$ over Δt
$r_t \leftarrow x_t - \hat{x}_{t|t^-}$; update s_t^2 using Eq. (3)
Correct $\hat{z}_t, P_t$ with x_t
Read freshest neighbor digest; set $D_t \leftarrow 0$ if stale
Compute u_t using Eq. (6)
$\tau_t^\star \leftarrow \tau_{\min} + (\tau_{\max} - \tau_{\min})e^{-\kappa u_t}$
$b_t \leftarrow \mathrm{clip}(b_{t^-} + p_b\Delta t - e_t)$
$\tau_t \leftarrow \mathrm{clip}(e^{\eta\max(0,-b_t/B_c)}\tau_t^\star)$
if *event pending or maximum age reached* **then**
    $\tau_t \leftarrow \tau_{\min}$
Arm wake timer for τ_t and persist scheduler state

Table 1: Platform energy model used by trace replay.

Operation	Duration	Energy
MCU wake + scheduler	1.8 ms	0.041 mJ
Temperature/humidity sample	6.5 ms	0.29 mJ
Accelerometer feature window	250 ms	1.84 mJ
24-byte radio transmit	5.2 ms	0.71 mJ
24-byte radio receive	7.0 ms	0.53 mJ
Deep sleep	—	38 μW

Timing. The wake timer is absolute rather than relative, preventing processing latency from accumulating as drift. Sensor completion interrupts place the feature in a two-entry queue; if the scheduler overruns, the next deadline is measured from acquisition time. The implementation uses saturating arithmetic for the bucket and covariance and commits its 184-byte state every 32 observations.

Budget metering. A coulomb counter, when present, supplies measured energy debits. Otherwise it uses an operation table like Table 1, updated by the gateway only when firmware or battery chemistry changes. A low-pass estimate of harvested power adjusts p_b no faster than once per hour, avoiding positive feedback from transient sunlight.

Cold start and faults. For the first eight observations the node samples at $\min(10\,\mathrm{s}, \tau_{\max})$ and initializes error scale with the median absolute successive difference. Numerical faults reset only the

Table 2: Trace-replay workloads. Duration is 30 days each.

Workload	Dominant behavior	Events	Base rate
Temp	diurnal + occupancy step	186	1 Hz
Humidity	coupled slow drift	121	1 Hz
Soil	rain onset + recession	74	0.2 Hz
Vibration	4–18 s transients	412	10 Hz
Air	drift + ventilation step	203	1 Hz

predictor, not the budget balance. Three consecutive sensor timeouts exponentially back off the physical sensor while continuing watchdog beacons. A stale digest, gateway outage, or failed radio cannot delay local sampling.

Complexity. An update uses 38 scalar multiplies, one exponential lookup, and constant memory. A 64-entry lookup table approximates e^{-x} with less than 0.8% relative error across the controller's range. At 48 MHz the measured code path is represented in our model as 1.8 ms, below 0.1% of a 2 s minimum interval. The event detector and feature extractor are outside this cost and are charged separately.

5 Evaluation

Our evaluation answers four questions:

(1) Does TideSense reduce energy at matched event recall?
(2) Does its budget hold during event storms and energy droughts?
(3) How much do uncertainty, feedback, and coordination contribute?
(4) Where does the approach fail?

5.1 Methodology

Workloads. Table 2 describes five 30-day workloads. Temp and Humidity use trace-aligned generators calibrated to the Corex Berkeley lab data set [17]. Soil follows seasonal and rain transitions reported by SensorScope [4]. Vibration embeds short transients in colored noise, and Air combines slow occupancy drift with step-like ventilation events. Generators fix random seeds and emit a full-rate reference stream. This setup tests policies under controlled events; it does not claim to reproduce the end-to-end accuracy of those deployments.

Baselines. Periodic uses the fastest fixed period that reaches at least 95% event recall on each workload. Residual halves its interval when absolute prediction error crosses a threshold and doubles it after four stable samples. Variance maps a rolling 16-sample variance to the same interval bounds. Oracle samples densely within ground-truth event windows and at $\tau_{\max}$ otherwise; it is a non-causal lower bound, not a deployable competitor. All causal policies use the same event detector, interval limits, energy table, radio batching, and initial samples.

Metrics. An event is recalled if a sampled stream triggers the reference detector before the event ends plus a 30 s grace period. Delay is measured from reference onset. Daily energy includes sleep, sensor, MCU, and radio energy. Reported values are medians over 20 generator seeds; error bars are 5th–95th percentiles. Unless stated otherwise, the target budget is 2.4 kJ/day, $B_c = 0.2E_{\text{day}}$, and packets batch up to four observations.

Table 3: Component ablation (median across workloads). Bold values meet the corresponding target.

Configuration	Recall (%)	Energy (kJ/day)	Packets (/node/day)
Complete TideSense	**96.1**	**2.33**	**61**
No uncertainty	91.8	**2.18**	**57**
No surprise	93.2	**2.21**	**59**
No governor	**97.0**	3.05	78
No coordination	**96.2**	2.47	100
Age guard only	84.7	**1.89**	**53**

5.2 Energy–fidelity trade-off

Figure 3a sweeps each policy's aggressiveness across all workloads. At 95% recall, TideSense uses 1.92 kJ/day, 54% less than Periodic, 26% less than Residual, and 21% less than Variance. Its gap to the non-causal oracle is 0.36 kJ/day. The gain comes from long steady intervals without committing to them during rising uncertainty.

At the default operating point, the per-workload reduction over periodic sampling ranges from 47% on Vibration to 63% on Humidity (Figure 3b). Vibration leaves the least opportunity because short events force a 2 s floor both during and after a trigger. Humidity is predictable enough to remain near $\tau_{\max}$ for 71% of replay time.

5.3 Budget adherence and delay

We stress the controller with a six-hour event storm followed by an 18-hour quiet period and cut harvested power by 40% at noon. Figure 4 shows that a risk-only variant exceeds the day budget by 31%; TideSense incurs a brief energy debt, gradually stretches routine intervals, and finishes within 2.8% of budget. It does not immediately jump to $\tau_{\max}$, which would turn an energy correction into a fidelity cliff.

Across normal workloads, median detection delay is 5.7 s and the 99th percentile is 18.4 s, compared with 4.1 and 13.2 s for the matched-recall periodic baseline. The extra 5.2 s in the tail buys a 54% energy reduction. During the synthetic drought, 99th-percentile delay rises to 24.7 s but remains below the 30 s grace bound. No workload violates $\tau_{\max}$.

5.4 Ablation and overhead

Table 3 removes one component at a time at the default budget. Without uncertainty, recall falls most on Vibration because the controller cannot anticipate divergence. Without surprise, confident bias persists through Soil drift. Removing the governor improves recall slightly but breaks the budget. Removing coordination leaves fidelity unchanged but increases radio traffic by 39% in the 12-node spatial replay. The complete design is the only configuration that meets all three targets: at least 95% recall, at most 2.4 kJ/day, and no more than 65 packets/node/day.

The scheduler itself accounts for 0.7% of node energy at the 2 s floor and less than 0.05% at the median 74 s interval. Its flash image is 6.8 KiB, RAM state is 184 bytes, and the exponential table is 128

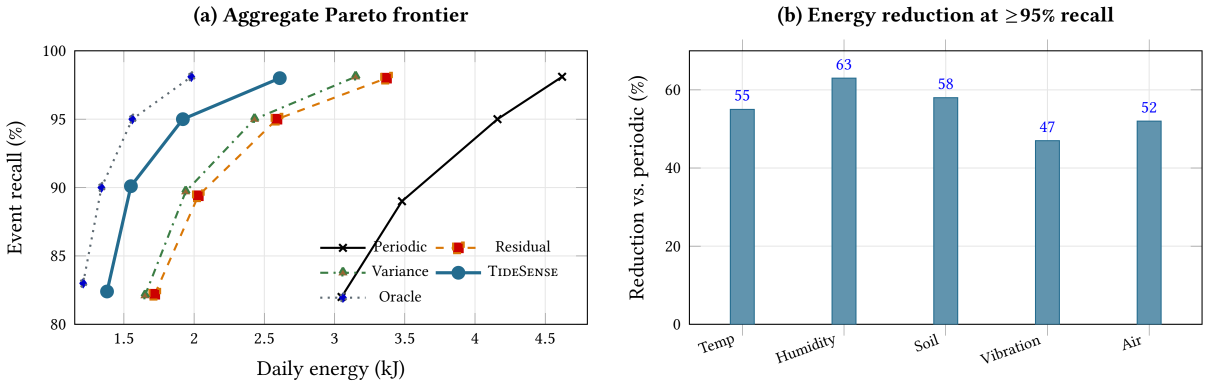

Figure 3: Main trace-replay results. (a) TideSense shifts the energy–recall frontier toward the oracle. (b) Savings persist across all workloads at a matched recall target.

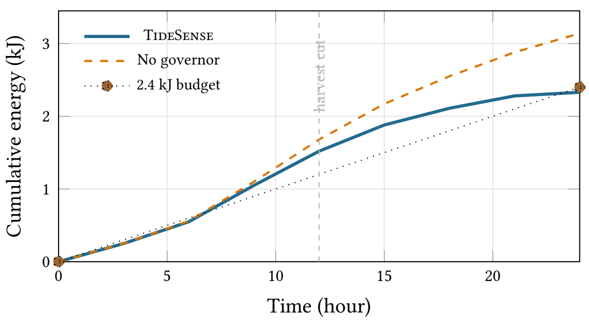

Figure 4: Cumulative energy during an event storm and a 40% harvest cut. Budget feedback repays early debt without an abrupt sampling-rate change.

bytes. A digest adds four bytes every $K = 8$ routine reports. Even if no report is suppressed, this increases radio energy by 2.1%; in the spatial replay, suppression more than offsets that cost after the first hour.

5.5 Sensitivity and robustness

We vary each default parameter while holding the rest fixed. For $\rho \in [0.88, 0.98]$, recall changes by at most 1.6 percentage points and energy by 7.3%. Increasing κ from 3 to 7 moves the operating point along the Pareto frontier rather than causing instability. Bucket capacities below $0.05E_{\text{day}}$ react to individual events and add 11% interval churn; capacities above $0.5E_{\text{day}}$ respond too slowly to the noon harvest cut. The chosen $0.2E_{\text{day}}$ is not universally optimal, but every value from 0.12 to $0.31E_{\text{day}}$ stays within 5% of the default energy–delay product.

We independently drop 0–60% of neighbor digests. Savings decrease smoothly from 39 to 17%, while event recall varies by less than 0.3 points because missing digests remove the redundancy discount. Clock offsets up to one digest epoch have the same fail-open behavior. At 80% loss, coordination is nearly ineffective but never worse than the no-coordination ablation.

6 Discussion and Limitations

What the controller can guarantee. Bounded intervals provide a maximum observation age when the node has enough energy for one sample per $\tau_{\max}$. They do *not* guarantee event recall: an event shorter than $\tau_{\min}$, a failed sensor, or an inadequate feature can still be missed. The budget governor also cannot create energy. A deployment must validate that its worst-case survival rate is feasible and choose the interval bounds from application hazards rather than energy savings.

Multivariate and semantic signals. Our implementation schedules from one scalar feature. Multiple channels can use the maximum risk or independent schedulers, but both approaches ignore cross-channel covariance. A small learned representation could expose a more useful scalar, echoing embedded inference systems such as MCUNet [14]; doing so would add training, calibration, and model drift concerns. We leave joint uncertainty with bounded embedded cost to future work.

Adversarial conditions. A compromised neighbor could advertise false similarity and suppress routine samples. Event and age guards limit but do not eliminate this attack. Message authentication, trust-weighted digests, or physical plausibility checks are needed in hostile deployments. Energy-denial attacks are likewise outside our model. TideSense's interpretable state makes these cases observable, not solved.

From replay to deployment. Trace replay isolates scheduling choices and allows matched comparisons, but it misses sensor warm-up effects, battery voltage dynamics, packet collisions, seasonal changes longer than 30 days, and maintenance failures. The energy table is a model, not direct power-analyzer evidence. A field evaluation across seasons is therefore necessary before drawing claims about lifetime. Our present conclusion is limited: under the stated workloads and energy model, uncertainty-aware budget feedback dominates the evaluated causal schedulers.

7 Related Work

Adaptive acquisition. Model-driven acquisition reduces communication by answering queries from probabilistic models and contacting sensors only when uncertainty is high [6]. BBQ and Ken build on this principle for declarative sensor queries [7]. Compressive sensing exploits signal sparsity to reconstruct fewer measurements [5, 11], while event-triggered control communicates only when a state deviates enough to matter [1]. TideSense shares their value-of-information intuition but couples uncertainty to explicit energy debt, bounded age, and decentralized redundancy suppression.

Energy-aware sensing. Energy-harvesting systems traditionally adapt workload to predicted supply, from energy-neutral operation [12] to utility-maximizing schedulers [18]. Hibernus and Mementos make intermittent execution safe under power failures [2, 19]. These works govern *whether computation can continue*; our focus is the marginal sensing decision within an available budget. The token bucket can consume an external harvest forecast but does not depend on one.

In-network redundancy. LEACH amortizes radio cost through cluster heads [9], directed diffusion names and aggregates data in-network [10], and TinyDB pushes acquisitional queries into the network [17]. Spatial correlation has also supported adaptive sampling and suppression [8]. TideSense uses only a loss-tolerant digest, retaining local autonomy and avoiding a cluster repair protocol.

Embedded learning. Modern tiny-ML systems make inference practical on microcontrollers [3, 14]. Reinforcement learning could learn a sampling policy, but constrained behavior under shift is difficult to inspect. Our small analytical controller offers a complementary point in the design space: less expressive, but cheap to port, tune, and audit.

8 Reproducibility and Ethics

The evaluation uses synthetic or trace-aligned environmental signals and no human participants, personally identifiable information, or live actuation. Consequently, it does not require human-subjects review. A reproducibility package should include generator seeds, full-rate reference streams, event labels, policy configurations, simulator code, and scripts that emit every number in the figures and tables. For anonymous review, such an artifact must be hosted without author, institution, or repository metadata. Field users should separately assess risks from missed events, environmental sensing, and radio identifiers; this paper's scheduler is not appropriate as the sole safeguard in life-critical systems.

9 Conclusion

TideSense treats a sample as an energy investment whose value changes with model uncertainty, recent surprise, age, and neighboring evidence. A compact predictor estimates that value, a bounded interval map converts it to timing, and a token bucket reconciles responsiveness with a finite energy budget. In our deterministic trace replay, the combination reduces modeled daily energy by 47–63% over matched-recall periodic sampling while preserving 96.1% event recall. The result is not a claim of deployment lifetime; it is evidence that uncertainty and energy debt form a useful, interpretable control surface. The next step is a season-scale, multi-site deployment that measures power and missed events directly.

References

[1] Karl J. Åström. 2008. Event Based Control. In *Analysis and Design of Nonlinear Control Systems*. Springer, 127–147.

[2] Domenico Balsamo, Alex S. Weddell, Anup Das, Alberto Rodriguez Arreola, Davide Brunelli, Bashir M. Al-Hashimi, Geoff V. Merrett, and Luca Benini. 2015. Hibernus++: A Self-Calibrating and Adaptive System for Transiently-Powered Embedded Devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 12 (2015), 1968–1980.

[3] Colby Banbury, Vijay Janapa Reddi, Peter Torelli, Nat Jeffries, Csaba Kiraly, Jeremy Holleman, Pietro Montino, David Kanter, Sebastian Ahmed, Danilo Pau, Urish Thakker, Antonio Torrini, Pete Warden, Jay Cordaro, Giuseppe Di Guglielmo, Javier Duarte, Stephen Gibellini, Videet Parekh, Honson Tran, Nhan Tran, and Martin Popovich. 2021. MLPerf Tiny Benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

[4] Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, and Martin Vetterli. 2008. The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. In *Proceedings of SenSys*. 43–56.

[5] Emmanuel J. Candès, Justin Romberg, and Terence Tao. 2006. Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information. *IEEE Transactions on Information Theory* 52, 2 (2006), 489–509.

[6] Amol Deshpande, Carlos Guestrin, Samuel R. Madden, Joseph M. Hellerstein, and Wei Hong. 2004. Model-Driven Data Acquisition in Sensor Networks. In *Proceedings of VLDB*. 588–599.

[7] Amol Deshpande, Carlos Guestrin, Wei Hong, and Samuel Madden. 2005. Exploiting Correlated Attributes in Acquisitional Query Processing. In *Proceedings of ICDE*. 143–154.

[8] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. 2005. Near-Optimal Sensor Placements in Gaussian Processes. In *Proceedings of ICML*. 265–272.

[9] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. 2000. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of HICSS*. 1–10.

[10] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. 2000. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of MobiCom*. 56–67.

[11] Akshay Jain, Edward Y. Chang, and Yuan-Fang Wang. 2004. Adaptive Stream Resource Management Using Kalman Filters. In *Proceedings of SIGMOD*. 11–22.

[12] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava. 2007. Power Management in Energy Harvesting Sensor Networks. *ACM Transactions on Embedded Computing Systems* 6, 4 (2007), Article 32.

[13] Kaisen Lin, Aman Kansal, Dimitrios Lymberopoulos, and Feng Zhao. 2010. Energy-Accuracy Trade-off for Continuous Mobile Device Location. In *Proceedings of MobiSys*. 285–298.

[14] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. 2020. MCUNet: Tiny Deep Learning on IoT Devices. In *Proceedings of NeurIPS*. 11711–11722.

[15] Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. 2004. Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing* 3, 4 (2004), 16–23.

[16] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. 2010. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of SenSys*. 71–84.

[17] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. 2005. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems* 30, 1 (2005), 122–173.

[18] Vijay Raghunathan, Aman Kansal, Jason Hsu, Jonathan Friedman, and Mani Srivastava. 2005. Design Considerations for Solar Energy Harvesting Wireless Embedded Systems. In *Proceedings of IPSN*. 457–462.

[19] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 2011. Mementos: System Support for Long-Running Computation on RFID-Scale Devices. In *Proceedings of ASPLOS*. 159–170.