Energy and QoS-Aware Dynamic Reliability Management of IoT Edge Computing Systems

Kazim Ergun
University of California, San Diego
kergun@ucsd.edu

Raid Ayoub
Intel Corporation
raid.ayoub@intel.com

Pietro Mercati
Intel Corporation
pietro.mercati@intel.com

Dancheng Liu
University of California, San Diego
d1liu@ucsd.edu

Tajana Rosing
University of California, San Diego
tajana@ucsd.edu

ABSTRACT

The Internet of Things (IoT) systems, as any electronic or mechanical system, are prone to failures. Hard failures in hardware due to aging and degradation are particularly important since they are irrecoverable, requiring maintenance for the replacement of defective parts, at high costs. In this paper, we propose a novel dynamic reliability management (DRM) technique for IoT edge computing systems to satisfy the Quality of Service (QoS) and reliability requirements while maximizing the remaining energy of the edge device batteries. We formulate a state-space optimal control problem with a battery energy objective, QoS, and terminal reliability constraints. We decompose the problem into low-overhead subproblems and solve it employing a hierarchical and multi-timescale control approach, distributed over the edge devices and the gateway. Our results, based on real measurements and trace-driven simulation demonstrate that the proposed scheme can achieve a similar battery lifetime compared to the state-of-the-art approaches while satisfying reliability requirements, where other approaches fail to do so.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; Sensor networks; *Reliability*.

KEYWORDS

Internet of Things, Edge Computing, Dynamic Reliability Management

ACM Reference Format:

Kazim Ergun, Raid Ayoub, Pietro Mercati, Dancheng Liu, and Tajana Rosing. 2021. Energy and QoS-Aware Dynamic Reliability Management of IoT Edge Computing Systems. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21)*, January 18–21, 2021, Tokyo, Japan. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3394885.3431612>

This work is licensed under a Creative Commons Attribution International 4.0 License.
ASPDAC '21, January 18–21, 2021, Tokyo, Japan
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7999-1/21/01.
<https://doi.org/10.1145/3394885.3431612>

1 INTRODUCTION

The Internet of Things (IoT) comprises billions of interconnected heterogeneous devices that have the ability to sense, communicate, compute, and actuate. Due to the shortcomings of traditional cloud-centric IoT (e.g., latency, energy, privacy, cost) [1, 2], Edge Computing [3] is emerging as a promising solution, where data processing is pushed to the edge of the IoT network. The edge devices will run heavy workloads, dissipate more power than ever, and heat up, with no active cooling. They operate in diverse and sometimes harsh environments, thus, are often subject to external (due to ambient temperature) as well as internal (due to power dissipation) temperature stress, bringing reliability concerns. For typical edge computing systems, as studied by prior work [4–7], improving the energy efficiency of devices while delivering a minimum Quality of Service (QoS) is the main goal since many edge devices are battery-operated or have limited energy sources [8]. As edge devices undertake bigger workloads, thermal stress and reliability issues cannot be neglected. To date, no prior work on edge computing considered reliability.

A significant portion of spending on the IoT (\$746 billion in 2019 [9]) is associated with maintenance and technical diagnostics due to system failures [10]. Among various system failures, hard failures in hardware, for which the devices age, degrade, and eventually fail, are particularly important since they are irrecoverable, requiring maintenance for the replacement of defective parts at high costs. Hard failures can be prevented or deferred by mitigating *reliability degradation*, to which we devote our attention to in this work. Reliability degradation is mainly induced by temperature stress, which depends on power dissipated for running workloads and environmental conditions, such as ambient temperature. Dynamic reliability management (DRM) techniques have been proposed in traditional devices such as personal computers, mobile devices, and data centers [11–13] to mitigate degradation by trading off performance for meeting reliability criteria. However, no work has been done to consider a network-level IoT reliability management framework.

Ideally, the gateway must be self-organizing and self-supported, handling the management of the edge computing system with no or minimum dependency on the cloud [1, 14]. This means that a light-weight, low-overhead, *dynamic* and *scalable* solution at the gateway is needed for the management to be responsive to variations in the system and handle large number of edge devices distributed over the network. However, the problem of managing the reliability of edge devices poses high complexity due to its

size and nonlinearity; it is infeasible to solve it with traditional methods such as centralized nonlinear model predictive control on resource-constrained gateways.

In this paper, we present a novel DRM technique for IoT edge devices, taking advantage of the Edge Computing architecture where a portion of the computation assigned to the edge devices can be offloaded to the IoT gateways. The goal of the management is to satisfy the QoS and reliability requirements while maximizing the remaining energies of the device batteries. To this end, we formulate a finite horizon nonlinear optimal control problem with a battery energy objective, QoS, and terminal reliability constraints. We then propose a hierarchical multi-timescale distributed controller solution to deal with the high complexity of the problem. We decompose the problem into low-overhead sub-problems that are solved by leveraging a cascade of linear controllers that act on different time scales, distributed over the edge devices and the gateway. Unlike the DRM techniques for stand-alone devices, our approach exploits both individual (dynamic voltage and frequency scaling) and network-level (offloading) controls to mitigate reliability degradation. To the best of our knowledge, we are the first to address the reliability management problem in a networked edge computing setting. Based on real measurements, trace-driven simulations demonstrate that our proposed scheme can achieve a similar battery lifetime compared to the state-of-the-art approaches while satisfying reliability requirements, where other approaches fail to do so.

2 RELATED WORK

Most IoT edge devices are powered with batteries, thus many works aim at balancing the tradeoff between power consumption and delay performance. The authors in [4] and [15] characterize the computation and communication energy and performance of data processing applications across edge devices and servers, then identify where to run the application. In [16], the same problem in a mobile-cloud computing setting is formulated as a non-convex optimization problem. The shortcoming of these approaches is that they only support two operation modes: entirely offloading the computation or entirely processing it locally. In this regard, a scheme for partitioning the computation of applications was employed to minimize energy consumption while satisfying a completion time requirement in [17] and [7]. However, the offload target is assumed to be very powerful and fast, or to have unlimited resources. Moreover, only one edge/mobile device is considered, without accounting for resource contention between the network devices.

In the edge computing architecture, there are limited resources (bandwidth, gateway's processing power) shared between multiple devices. Prior work on computation offloading for edge computing examined either the allocation of distinct tasks or different stages of applications to edge devices and gateways [18–20]. These problems are commonly formulated as Integer Linear Programming problems and solved with heuristics to find the best allocation of application stages/tasks, from a finite set of options. The application tasks that are selected by the aforementioned techniques can be used as an input to our problem. Assuming prior allocation, we find the optimal *rate of input data* to be processed locally at the edge and to be offloaded to the gateway. Different from previous studies,

Figure 1: Structure of the edge device and network operation

we have a control-theoretic approach; we treat the selection of processing and offloading rates as an optimal control problem.

Extensive literature exists for the reliability degradation phenomena on system-on-chips (SoCs). In these works, physical-level models are built to quantify the reliability degradation due to voltage and temperature stress, which are influenced by the environmental conditions and workload variations. Based on the reliability models, a management algorithm optimizes performance while satisfying reliability constraints. The trade-off between performance and reliability can be adjusted during runtime by power/voltage scaling [21–24], task scheduling [25], or both [12]. In [26], a task allocation scheme is presented for multi-processor SoCs which maximizes the time to failure of an SoC subject to performance constraints. The authors in [12] implement the above-mentioned mechanisms on a mobile device, showing as much as a one-year improvement on lifetime with dynamic reliability management.

Despite the impressive results on individual devices, reliability management for networks of IoT devices is an open problem. The recent paper in [8] briefly discussed reliability in the context of IoT and acknowledged that IoT devices can profit from voltage scaling with respect to power and energy. To the best of our knowledge, we are the first to propose reliability management for edge computing, which leverages both individual controls (voltage/frequency scaling) and network-level mitigation strategies, such as computation offloading.

3 FRAMEWORK ARCHITECTURE

In this section, we first present the system model and describe the operation of the edge devices and the gateway. Subsequently, we give the problem statement and propose a dynamic reliability management technique that implements a novel hierarchical multi-timescale control.

In many IoT edge computing systems, the application is not entirely executed on a single device, instead, it is segmented into *tasks* and distributed over computing hierarchy, consisting of the cloud, the fog, and the edge devices [3]. In our work, we assume that this segmentation and distribution process is governed by an external management mechanism, such as [18]. Therefore, the edge devices in our network are dynamically being assigned tasks. As shown in

Figure 1, the tasks can be executed either locally at the edge devices or remotely on the gateway node via computation offloading. In particular, the *input data* of the tasks can be partitioned and offloaded (communicated) to the gateway. In the case of offloading, both the edge device and the gateway execute the same task, but at different times and on different partitions of the data. The application code is assumed to be already present on both devices. To characterize a task τ_m , we consider three attributes: $\{IPC_m, \alpha_m, D_m\}$. Here, IPC_m is the average instruction per cycle required to run the task, α_m is the activity factor, and D_m represents the deadline. The input of the tasks are data generated by the sensors of the edge device, which produces traffic with an arrival rate λ_i . We use index i to uniquely identify edge device ED_i .

The operation of an edge device is illustrated in Figure 1, with local data processing at its processing unit (PU) and network communication for data offloading at its transceiver (RF). The rate at which the traffic is forwarded to the gateway through RF is L_i , denoted as the offloading rate. The computation intensity (processing rate), $\mu_i(f_i, \tau_m)$, is deterministic and dependent on the edge device's operating clock frequency and the running task (due to its IPC). Both the PU processing rates μ_i and RF communication rates L_i are controllable variables that are regulated by our proposed DRM controller. We assume the communication of task outputs is negligible, but the proposed models can be extended to account for it.

Data from the edge device is sent to the gateway through a wireless channel with bandwidth BW . The gateway receives the superposition of offloaded periodic traffics from a number of unsynchronized network devices. According to the Palm-Khintchine theorem, this aggregated traffic can be approximated with a Poisson process with the arrival rate $\sum_{i=1}^N L_i$, where N is the number of edge devices [27]. The computing resource of the gateway is adequate for processing data for several tasks from multiple edge devices simultaneously. We assume that there are c homogeneous computation cores in the gateway's SoC, working with a deterministic processing rate μ_G . Also, unlike the edge devices, memory resource of the gateway is sufficient to be able to hold a queue of incoming data. Therefore, the gateway employs a queueing structure of type M/D/c [28], denoted QG .

The edge devices are battery-powered with limited capacities q_i . They consume the power amount $P_{sen,i}$ for sensing, $P_{pu,i}$ for computation, and $P_{rf,i}$ for communication. We define the power consumption vector of the edge device, $P_{ed,i} = [P_{pu,i}, P_{rf,i}]^T$, only including the two controllable terms. Accordingly, the internal temperature $T_{ed,i}$ of the device i (involving PU and RF components) is influenced by its power dissipation and by the ambient temperature $T_{amb,i}$ of the environment. Finally, $R_{ed,i}$ denotes the reliability, which is heavily affected by temperature $T_{ed,i}$. We assume that the gateway is placed at a much more central location than the edge devices, thus, (i) it can have its power supplied by the grid and (ii) reliability concerns are not as significant (e.g. access to cooling).

3.1 Proposed Control System

We propose a novel dynamic control system to maximize the remaining battery energy of the edge devices under reliability and QoS constraints. This is a high dimensional nonlinear problem where the nonlinearity comes mainly from the cubic frequency-power

Figure 2: Controller block diagram

relation [29]. Approaches such as centralized nonlinear model predictive control are not scalable to this class of problems. To address the scalability hurdle, we decompose the full control problem into subproblems with coordination by leveraging a hierarchy of linear controllers that act on different time scales, distributed over the edge devices and the gateway. The goal of this decomposition is twofold: first, to ensure that the gateway's subproblem is computationally less intensive and smaller in size, and second, to ensure that the coupling between local subproblems is minimal and solvable in tolerable time on constrained edge devices.

Figure 2 shows our *hierarchical multi-timescale* control approach. The lower level controllers at each edge device manage the local, 'decoupled' variables, whereas the top-level controller at the gateway coordinates the control decisions among the controllers at the lower level. The overall system consists of subsystems whose control variables operate at different time scales. Leveraging this, we apply three different time scales: Long Intervals (LI), in the order of hours that targets slow reliability changes, Medium Intervals (MI), in the order of seconds for temperature variations, and Short Intervals (SI), in the order of milliseconds for performance-related decisions. In this multi-timescale approach, the faster-varying subsystems arrive at their steady-state before the slower subsystems, which minimizes violations; thus, it leads to a minimal loss in control quality with a significant reduction in complexity [30]. The proposed controller architecture consists of the following four components.

1) Reliability Controller: Estimates the reliability degradation of the edge devices at the beginning of each LI. Based on the current reliability value and the target reliability constraint R^{target} , it computes a reference temperature T_{ed}^{ref} , which is used as a constraint by the *Edge Thermal Controller*.

2) Edge Thermal Controller: Computes the maximum reference power dissipation value P_{ed}^{ref} , which would ensure that, at the end of the LI timescale, the average temperature experienced in the whole LI is less than T_{ed}^{ref} . Then, it modifies these maximum values based on the target input data rate to obtain a lower, *energy optimal* power reference, which is sent to the *Gateway Top-Level Controller*.

3) Gateway Top-Level Controller: Calculates the reference optimal communication rates L^{ref} for each edge device (at each MI timescale) that maximizes their remaining battery energies and satisfy the delay requirements of their respective tasks, while abiding by the bandwidth limit BW .

4) Edge Performance Controller: Computes the edge device computation and communication rates by applying controls Δf and ΔL at each SI time scale.

3.2 Edge Reliability Controller

The reliability R_{ed} of an edge device (its PU and RF) at the end of a predefined lifetime t_{life} should be greater than a target reliability R^{target} . Reliability degradation is induced by mechanisms such as Time Dependent Dielectric Breakdown, Negative Bias Temperature Instability, Hot Carrier Injection, and Electromigration, which all show a strong dependence on temperature. The reliability function $R(t)$ of these mechanisms, in general, is expressed as a monotonically decreasing exponential function of time and temperature [21].

$$R(t) = \gamma_1 \exp\left(-\frac{E_a}{kT}\right) \exp(-\gamma_2 t) \quad (1)$$

where E_a is the activation energy, k is Boltzmann's constant, and γ_1 , γ_2 are the constants depending on the respective mechanism. The expression in Equation (1) is the function depicting the probability of not having failures before a given time t , defined in the interval $[0,1]$. It is only representative of static systems because it assumes a constant temperature applied from time $t = 0$. The workloads and temperature vary over time, so is the degradation process. Therefore, we introduce *equivalent degradation time* to characterize the reliability degradation effect under such varying conditions. Given the reliability degradation of a device under temperature T_1 for duration t_1 , the equivalent degradation under temperature T_2 is described as follows:

$$\Delta R(t_{eqv,1}, T_2) = \Delta R(t_1, T_1) \quad (2)$$

The equivalent degradation time $t_{eqv,1}$ can be computed using Equation (1). To elaborate, assume a scenario where a device worked subsequently under temperature T_1 and T_2 , with durations t_1 and t_2 , respectively. Then, the degradation of the device at time $t_1 + t_2$ equals that of the device which worked under temperature T_2 for time $(0, t_{eqv,1} + t_2)$, and can be computed as $\Delta R(t_{eqv,1} + t_2, T_2)$.

To capture the dynamics of reliability under varying temperature, we discretize the time and calculate reliability at each time step as shown in the following. We leverage the equivalent degradation time to calculate the degradation at each discrete time step. The temperature is assumed to be constant between time steps.

$$\begin{aligned} \Delta R(t_{eqv,k-1}, T_{k-1,k}) &= R_0 - R_d(k-1) = \Delta R|_{t=t_{k-1}} \\ R_d(k) &= R(t_{eqv,k-1} + t_{k-1,k}, T_{k-1,k}) \end{aligned} \quad (3)$$

In Equation (3), k indicates the k^{th} time instant and $T_{k-1,k}$ is the temperature experienced by the device between the time instants $k-1$ and k . Similarly, $t_{k-1,k}$ is the time passed between the time instants $k-1$ and k . R_d is the dynamic reliability and R_0 is the reliability of a device at time $t = 0$.

Leveraging the above equivalent degradation time technique, the *Edge Reliability Controller* calculates the reliability degradation of the edge device at each LI, using the averaged temperature over the previous LI. Then, it selects the reference temperature T_{ed}^{ref} for the next LI by solving the convex optimization problem in Equation (4). Since the controller activates by intervals in the order of days, the computation of convex optimization introduces a negligible overhead.

$$\begin{aligned} \min_{T_{ed}^{ref}} \quad & \|R(t_{eqv} + t_{rem}, T_{ed}^{ref})\| \\ \text{s.t.} \quad & R(t_{eqv} + t_{rem}, T_{ed}^{ref}) \geq R^{target} \\ & R(t_{eqv}, T_{ed}^{ref}) = R_{d,ed}(k_{LI}) \end{aligned} \quad (4)$$

$R_{d,ed}(k_{LI})$ indicates the dynamic reliability at the long interval k_{LI} and t_{rem} is the remaining time from the current LI until t_{life} . The result of the optimization, T_{ed}^{ref} , is the temperature which would satisfy the reliability target R^{target} at the desired lifetime t_{life} .

3.3 Edge Thermal Controller

Within a long interval, the *Edge Thermal Controller* determines the power $P_{ed}(k_{MI})$ at each MI time step k_{MI} , which would ensure that the temperature T_{ed} experienced in the whole LI on average is less than T_{ed}^{ref} . The discrete-time state-space model of the device's thermal behavior is expressed in the following equation [31].

$$T_{ed}(k_{MI} + 1) = A_T \cdot T_{ed}(k_{MI}) + B_T \cdot P_{ed}(k_{MI}) + C_T \cdot T_{amb}(k_{MI}) \quad (5)$$

A_T and B_T are defined as the state and the input matrices respectively. T_{amb} is the ambient temperature and C_T is a vector of coefficients which weighs the impact of ambient temperature on device's internal temperature. We apply the state-feedback control law [32] for a linear system with input reference to the Equation (5) in order to compute P_{ed} as follows:

$$P_{ed}(k_{MI}) = K_T (T_{ed}^{ref} - T_{ed}(k_{MI})) - C_T / B_T \cdot T_{amb}(k_{MI}) \quad (6)$$

where K_T is the feedback gain which is determined using pole placement technique. The ambient temperature T_{amb} is assumed to be known since it can be monitored with temperature sensors. If an edge device dissipates the resulting power P_{ed} at each time step k_{MI} , then it can very closely meet the reliability target R^{target} . However, to meet the QoS requirements on data rate λ^{target} , the edge device may need to consume more power than P_{ed} . Or, if λ^{target} is a relatively small rate, then consuming P_{ed} would be 'excess'. Therefore, to compute the reference power values P_{ed}^{ref} to be sent to the *Gateway Top-Level Controller*, we do a slight modification (trimming) on the power values P_{ed} obtained by Equation (6) concerning the power scaling of the components P_{pu} and P_{rf} .

The power consumption P_{pu} of a processing unit can be modeled as the sum two contributions: leakage power $P_{lea} = V_{dd}(bT^2e^{\frac{k}{T}} + I_{gate})$ and dynamic power $P_{dyn} = C^{eff}V_{dd}^2f$. Since the clock frequency f depends linearly on voltage V_{dd} [29], a simplified model that accounts for both dynamic and leakage power can be given as $P_{pu} = af^3 + bf$. On the other hand, the communication power consumption is determined by the rate of the bits transmitted over the wireless channel. It is characterized by the empirical transmission power model [33], and the required power P^{rf} to transmit L bits/second is $P^{rf} = \rho_1 \frac{L}{g} + \rho_2$. Here, ρ_1 denotes the energy coefficient, g denotes channel state, and ρ_2 is the constant power consumed by RF circuits.

P_{ed}^{ref} Trimming. Up until a certain rate, processing data on the PU consumes less power than communicating the data. However, it is more energy efficient to communicate data for higher rates since

Algorithm 1: Power Reference Trimming

Input: $\lambda^{target}, P_{pu}(k_{MI}), P_{rf}(k_{MI})$
Output: P_{ed}^{ref}

- 1 Calculate P_{pu} for $\mu = \lambda^{target}$
- 2 Calculate P_{rf} for $L = \lambda^{target}$
- 3 **if** $P_{pu}|_{\mu=\lambda} < P_{rf}|_{L=\lambda}$ **then**
- 4 **if** $P_{pu}|_{\mu=\lambda} < P_{pu}(k_{MI})$ **then**
- 5 $P_{ed}^{ref} = [P_{pu}|_{\mu=\lambda}, 0]^T$
- 6 **else**
- 7 Calculate $\{\mu^{ref} \mid P_{pu}|_{\mu=\mu^{ref}} = P_{pu}(k_{MI})\}$
- 8 $L^{ref} = \lambda^{target} - \mu^{ref}$
- 9 $P_{ed}^{ref} = [P_{pu}(k_{MI}), P_{rf}|_{L=L^{ref}}]^T$
- 10 **else**
- 11 **if** $P_{rf}|_{\mu=\lambda} < P_{rf}(k_{MI})$ **then**
- 12 $P_{ed}^{ref} = [0, P_{rf}|_{\mu=\lambda}]^T$
- 13 **else**
- 14 Calculate $\{L^{ref} \mid P_{rf}|_{L=L^{ref}} = P_{rf}(k_{MI})\}$
- 15 $\mu^{ref} = \lambda^{target} - L^{ref}$
- 16 $P_{ed}^{ref} = [P_{pu}|_{\mu=\mu^{ref}}, P_{rf}(k_{MI})]^T$

PU power consumption scales *superlinearly* with processing rate while RF power consumption scales *linearly* with communication rate. For a given λ^{target} , we use Algorithm 1 to find a tighter, more energy efficient power reference than what we have obtained in Equation (6) for the *Gateway Top-Level Controller's* problem. First, the algorithm calculates the required power consumption for processing or communicating data at rate λ^{target} . If processing power P_{pu} is the smaller of the two, then it is further compared with the value obtained from Equation (6). All needed power can be allocated to the PU if the required processing rate results in a power which is lower than what the thermal constraints allow (Line 5). Otherwise, we calculate the corresponding maximum allowed processing rate (μ^{ref}) and allocate the RF power consumption such that the rest of the data is communicated at rate L^{ref} (Lines 7-9). If initially processing power P_{rf} is found to be smaller, then same procedures are done for the RF (Lines 10-16).

3.4 Gateway Top-Level Controller

The goal of the *Gateway Top-Level Controller* is to maximize the remaining battery energies of the edge devices while assuring that the task delay requirements are satisfied and the bandwidth limit is not exceeded. The discrete queue dynamics at the input of the gateway is as follows:

$$Q_G(k_{MI} + 1) = [Q_G(k_{MI}) + \sum_{i=1}^N L_i(k_{MI}) - \mu_G(k_{MI})]^+ \quad (7)$$

In Equation (7), $Q_G(k_{MI})$ denotes the queue length of gateway at time instant k_{MI} , in bits, and $[x]^+ = \max(x, 0)$. μ_G is the total computation resources available at the gateway, in bits/s. We assume that the amount of μ_G can dynamically change depending on the overall network operation and we do not have control over it. We

exploit the delay aware virtual queue based on the ϵ -persistent queue technique [34]. It suggests that if the system can be controlled to ensure that $Q_G(k_{MI}) < Q_G^{max}$, then any task is fulfilled with a network delay smaller than its deadline, i.e. $D_G(k_{MI}) < D_m$. In addition to the delay requirement, the overall network utilization of the edge devices $\sum_{i=1}^N L_i(k_{MI})$ should not exceed the channel bandwidth BW .

To model the available energy in batteries, we use *Temperature Dependent Kinetic Battery Model* (T-KiBaM) [35]. T-KiBaM models the batteries with two tanks, respectively the Bound Charge Tank (BCT) and the Available Charge Tank (ACT). The ACT holds the electrical charge that can be immediately supplied to the load, while the BCT holds a secondary charge flowing towards the ACT. The battery is denoted empty when its ACT depletes. Let $P_{ov} = P_{pu} + P_{rf} + P_{sen}$ be the overall power drawn from the battery under supply voltage V_{dd} and q_A, q_B denote the total charge in ACT and BCT respectively. Their sum, $q_A + q_B$, is the total available charge in the battery. Equations (9a-b) give the system of equations that describes T-KiBaM.

Under these models, the *Central Gateway Controller* solves a standard quadratic programming (QP) form MPC with a linear system model.

$$\min_{\mathbf{P}_{ed}} \sum_{k_{MI}=0}^{T_p-1} -\|1^T \mathbf{q}(k_{MI})\|_Y^2 + \|\mathbf{P}_{ed}(k_{MI}) - \mathbf{P}_{ed}^{ref}\|_Z^2 \quad (8)$$

$$\text{s.t. } k_{MI} = 0, \dots, T_p$$

$$\begin{aligned} q_{A,i}(k_{MI} + 1) &= A_q q_{A,i}(k_{MI}) + B_q q_{A,i}(k_{MI}) - \frac{1}{V_{dd}} 1^T P_{ed,i}(k_{MI}) \\ q_{B,i}(k_{MI} + 1) &= C_q q_{A,i}(k_{MI}) + D_q q_{B,i}(k_{MI}) \end{aligned} \quad (9a-b)$$

$$Q_G(k_{MI} + 1) = [Q_G(k_{MI}) + \sum v^T P_{ed,i}(k_{MI}) - \mu_G(k_{MI})]^+ \quad (10)$$

$$Q_G(k_{MI}) \leq Q_G^{max} \quad (11)$$

$$\sum v^T P_{ed,i}(k_{MI}) \leq BW \quad (12)$$

where we used the fact that $[0 \text{ g}/\rho] \cdot P_{ed} = L$ and rewritten the communication rate variables L in terms of power variables P_{ed} with $v^T = [0 \text{ g}/\rho]$. $\mathbf{q} = [q_A, q_B]^T$ and $\mathbf{P}_{ed}$ are the combined vectors of all edge devices for battery states and power states respectively. Y and Z are matrices that weigh the importance of the elements in the cost function. Since the power reference vector $\mathbf{P}_{ed}^{ref}$ is constructed in the *Edge Thermal Controller* to yield an energy efficient reference, the two terms in the cost function are not conflicting. At each sampling time k_{MI} , the solver yields the optimal solution $\mathbf{P}_{ed}(k_{MI})$ within the MPC prediction horizon T_p that minimizes the cost function and meets the constraints. After converting power values to communication rate values, the respective communication rate references L_i^{ref} for each edge device are sent to the *Edge Performance Controllers*.

3.5 Edge Performance Controller

The two performance-related state variables controlled by the *Edge Performance Controller* are: (i) PU processing rate $\mu(f)$ and (ii) RF communication (offloading) rate L . Then, the control variables include the required change in the operating frequency Δf and

change in the communication rate ΔL . The *Edge Performance Controller* receives the reference communication rates $L^{ref}(k_{MI})$ from the *Gateway Central Controller* at each MI. Based on this, it adjusts $\Delta f(k_{SI})$ and $\Delta L(k_{SI})$ such that the sum of data processed locally and data offloaded amount to the data arrival rate $\lambda^{target}(k_{MI})$. They are computed using a similar state-feedback control law as in the *Edge Thermal Controller*.

$$\Delta f(k_{SI}) = K_{P,1}[(\lambda^{target}(k_{MI}) - L^{ref}(k_{MI})) - f(k_{SI})] \quad (13)$$

$$\Delta L(k_{SI}) = K_{P,2}(L^{ref}(k_{MI}) - L(k_{SI})) \quad (14)$$

4 EVALUATION

4.1 Experimental Setup

To illustrate the effectiveness of our solution, we conduct experiments on realistic edge computing scenarios. In our experiments, we use real power and temperature measurements collected from actual IoT devices.

Hardware: The target edge devices are Raspberry Pi 2 with ARM Cortex-A7 CPU and the gateway is Raspberry Pi 4 Model B with Arm Cortex-A72 CPU. The gateway (Pi 4) exhibits around ten times more instructions per second performance compared to edge devices (Pi 2) [36]. We model the CPU and WiFi power consumption and temperature of the edge devices by collecting measurements of various applications under different ambient temperatures. We use the *T-KiBaM* [35] as our battery model to realistically capture the discharge characteristics of the batteries. For reliability analysis, we calibrate the parameters of the model in Equation (1) by selecting a worst-case, a nominal, and a best-case device operation temperature (throughout its lifetime). These values are selected to be 70°C, 45°C, and 20°C respectively. We test our proposed technique with trace driven network simulations, following the characteristics of the modeled platform.

Environment: The reliability heavily depends on the temperature of the environment that the device operates, so we consider various ambient temperature conditions. We use the temperature dataset from [37], which contains hourly ambient temperature measurements of 36 cities for 5 years from 2012 to 2017. To demonstrate the effect of ambient temperature, we simulate scenarios in very hot (e.g., Phoenix) and cold (e.g., Toronto) locations. Moreover, we consider the effects of the device being placed in different places by selecting the temperature as $T_{amb} \pm U(-10, +10)$, where U is a uniform distribution. For example, a device placed in a closed container, when airflow around the device is restricted, so its heat is trapped, and the container is in the sun, will have much higher ambient temperature than a device placed under a shade in open air.

Application Scenario: Tasks assigned to the edge devices can be any segment of an application's pipeline. For example, traditional ML applications can be hierarchically segmented into filtering, feature extraction, and classification tasks. Or, neural networks (NN) can be inherently segmented into layers that have different jobs (e.g., convolutional layers for feature extraction in CNNs). In our experiments, we consider the ML classification and regression tasks characterized for edge computing settings in [4] with their corresponding power consumptions. These classification and regression tasks can either run on the edge devices or the gateway.

Figure 3: The relative time until edge device violates the reliability target of 0.85 (positive is better)

We conduct experiments based on a practical scenario of human activity recognition (HAR) [38], implemented on edge devices [39]. The task is to infer the label for one of five everyday activities (e.g. walking, running, cleaning, etc.) at the edge device using data gathered from three IMU sensors mounted at the chest, ankle and wrist, along with a heart rate monitor. The input rates from sensors to the classification task change due to (i) varying number of inference requests per second (QoS requirement), (ii) differing sampling rates of sensors based on desired signal quality [39].

4.2 Results

We compare our proposed approach with the following techniques:

- *No ERC-ETC* is our solution without the *Edge Reliability Controller* and *Edge Thermal Controller*.
- *All Edge* assigns all the computation to the edge devices with no offloading to the gateway.
- *Round Robin* is a method where the edge devices take turn offloading data.
- *Samie* (the name of the author) is the work presented in [6]. At each iteration of the algorithm, it finds the edge devices with the lowest and the second lowest battery life. Then, if their lifetimes can be extended by increasing their offloading, the edge devices are allocated more bandwidth.
- *Pagliari* (the name of the author) [15] makes the decision for offloading based on a combined metric of energy consumption and execution time demands of the tasks.

Reliability and Temperature: We first analyze the reliability benefits of adopting our solution for a network with 8 devices. The target reliability for the edge devices is selected to be 0.85 at t_{life} of 3 years (36 months). Figure 3 shows the time it takes for the edge device with the *minimum* reliability in the network to reach the target reliability of 0.85. The results are presented relative to the target lifetime of 36 months. Our approach reaches the target reliability at 37.7 months, whereas all other approaches fall short of the target lifetime. Figure 4 shows the reliability curve and temperature vs time for the edge device with the minimum reliability for our approach. The target reliability is met very closely at 3 years. As seen from the plots, the *Edge Thermal Controller* outputs a lower reference temperature when the average internal temperature of the device increases due to the varying ambient temperature.

Energy: Figure 5 presents the *minimum* remaining battery energy in the network. The values are plotted relative to the *All Edge* approach which consumes the most energy out of all methods. Our approach shows similar quality in terms of energy efficiency compared to the relative approaches while meeting the reliability

Figure 4: Reliability curve and the long time temperature behaviour of the device and the controller

Figure 5: Minimum battery charge in the network

requirements. *Samie* displays good results because their algorithm is tuned for energy savings, at each iteration of their algorithm, it specifically tries to improve the energy consumption of the device with the lowest battery energy. However, *Samie*'s approach violated reliability target by more than 15 months.

Quality of Service: The proposed controller satisfies input data rate, bandwidth, and deadline constraints. There are no violations due to the strict constraints in the MPC controller employed.

5 CONCLUSION

In this paper, we proposed a novel dynamic reliability management technique for IoT edge computing systems. We formulated a state-space control problem for QoS-aware and reliability-aware remaining energy maximization and solved it by leveraging a hierarchy of linear controllers that act on different time scales, distributed over the edge devices and the gateway. Our results show that the proposed scheme can achieve a given reliability target over its lifetime, in which the other relevant approaches fail.

ACKNOWLEDGMENTS

This work was partially supported by SRC task #2805.001, NSF grants #1911095, #1826967, #1730158 and #1527034, and by KACST.

REFERENCES

- [1] Zhang *et al.*, "The cloud is not enough: Saving iot from the cloud," in *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.
- [2] Hu *et al.*, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, 2017.
- [3] Shi *et al.*, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [4] Cui *et al.*, "Cross-Platform Machine Learning Characterization for Task Allocation in IoT Ecosystems," in *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–7.
- [5] Sheng *et al.*, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 114–126, 2015.
- [6] Samie *et al.*, "Computation offloading and resource allocation for low-power iot edge devices," in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016.
- [7] Kwak *et al.*, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.
- [8] Henkel *et al.*, "Ultra-low power and dependability for iot devices (invited paper)," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.
- [9] International Data Corporation, "The Growth in Connected IoT Devices," <https://www.idc.com/getdoc.jsp?containerId=prUS45213219>, 2019, [Online].
- [10] Cisco Jasper, "The hidden costs of delivering iiot services: Industrial monitoring & heavy equipment," Apr. 2016.
- [11] Karl *et al.*, "Multi-mechanism reliability modeling and management in dynamic systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2008.
- [12] Mercati *et al.*, "Warm: Workload-Aware Reliability Management in Linux/Android," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, 2016.
- [13] Xu *et al.*, "Dual: Reliability-aware power management in data centers," in *IEEE/ACM international symposium on cluster, cloud, and grid computing*, 2013.
- [14] Samie *et al.*, "Fast operation mode selection for highly efficient iot edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 3, 2019.
- [15] Pagliari *et al.*, "Input-dependent edge-cloud mapping of recurrent neural networks inference," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.
- [16] Sardellitti *et al.*, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.
- [17] Deng *et al.*, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE internet of things journal*, 2016.
- [18] Katsaragakis *et al.*, "Dmr: Distributed market-based resource management of edge computing systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1391–1396.
- [19] Liu *et al.*, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.
- [20] Cao *et al.*, "Qos-adaptive approximate real-time computation for mobility-aware iot lifetime optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, 2018.
- [21] Rosing *et al.*, "Power and reliability management of socs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 391–403, 2007.
- [22] Srinivasan *et al.*, "The Case for Lifetime Reliability-Aware Microprocessors," in *ACM SIGARCH Computer Architecture News*, vol. 32, no. 2, 2004.
- [23] Karl *et al.*, "Reliability Modeling and Management in Dynamic Microprocessor-Based Systems," in *Design Automation Conference*, 2006.
- [24] Zhuo *et al.*, "Process Variation and Temperature-Aware Reliability Management," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2010.
- [25] Coskun *et al.*, "Temperature Aware Task Scheduling in MPSoCs," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.
- [26] Huang *et al.*, "Lifetime reliability-aware task allocation and scheduling for mpsoic platforms," in *Design, Automation & Test in Europe Conference & Exhibition*, 2009.
- [27] Metzger *et al.*, "Modeling of aggregated iot traffic and its application to an iot cloud," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 679–694, 2019.
- [28] Kleinrock, *Queueing systems, volume 2: Computer applications*. Wiley, 1976.
- [29] Singla *et al.*, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 960–965.
- [30] Lunze, *Feedback control of large scale systems*. Prentice Hall PTR, 1992.
- [31] Beneventi *et al.*, "An Effective Gray-Box Identification Procedure for Multicore Thermal Modeling," *IEEE Transactions on Computers*, vol. 63, no. 5, 2012.
- [32] Franklin *et al.*, *Digital control of dynamic systems*. Addison-wesley Reading, MA, 1998, vol. 3.
- [33] Abo-Zahhad *et al.*, "An energy consumption model for wireless sensor networks," in *5th International Conference on Energy Aware Computing Systems & Applications*. IEEE, 2015.
- [34] Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011.
- [35] Rodrigues *et al.*, "A temperature-dependent battery model for wireless sensor networks," *Sensors*, vol. 17, no. 2, 2017.
- [36] MagPi, "The official Raspberry Pi Magazine," <https://magpi.raspberrypi.org/articles/raspberry-pi-specs-benchmarks>, 2019, [Online].
- [37] Kaggle, "Historical Hourly Weather Data," <https://www.kaggle.com/selfishgene/historical-hourly-weather-data>, 2017, [Online].
- [38] Reiss *et al.*, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers*. IEEE, 2012.
- [39] Thomas *et al.*, "Hierarchical and distributed machine learning inference beyond the edge," in *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, 2019.