Dynamic Reliability Management of Multi-Gateway IoT Edge Computing Systems

Kazim Ergun, *Student Member, IEEE,* Raid Ayoub, *Member, IEEE,* Pietro Mercati, *Member, IEEE,* and Tajana Rosing, *Fellow, IEEE*

***Abstract*—The emerging paradigm of edge computing envisions to overcome the shortcomings of cloud-centric Internet of Things (IoT) by providing data processing and storage capabilities closer to the source of data. Accordingly, IoT edge devices, with the increasing demand of computation workloads on them, are prone to failures more than ever. Hard failures in hardware due to aging and reliability 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 multi-gateway IoT edge computing systems to mitigate degradation and defer early hard failures. Taking advantage of the edge computing architecture, we utilize gateways for computation offloading with the primary goal of maximizing the battery lifetime of edge devices, while satisfying the Quality of Service (QoS) and reliability requirements. We present a two-level management scheme, which work together to (i) choose the offloading rates of edge devices, (ii) assign edge devices to gateways, and (iii) decide multi-hop data flow routes and rates in the network. The offloading rates are selected by a hierarchical multi-timescale distributed controller. We assign edge devices by solving a bottleneck generalized assignment problem (BGAP) and compute optimal flows in a fully-distributed fashion, leveraging the subgradient method. Our results, based on real measurements and trace-driven simulation demonstrate that the proposed scheme can achieve a similar battery lifetime and better QoS compared to the state-of-the-art approaches while satisfying reliability requirements, where other approaches fail by a large margin.**

***Index Terms*—Edge computing, computation offloading, constrained devices, device management, optimization and control.**

I. Introduction

The Internet of Things (IoT) comprises billions of interconnected heterogeneous devices that have the ability to sense, communicate, compute, and actuate. IoT continues to rapidly develop as it is adopted progressively across industries, in governments, and in consumers' daily lives. The number of interconnected IoT devices has already exceeded 10 billion and by 2025 it is expected to reach 40 billion [1]. A significant portion of spending on the IoT ($746 billion in 2019 [2]) is associated with maintenance and technical diagnostics due to system failures, which motivates our work.

Kazim Ergun is with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: kergun@ucsd.edu)

Raid Ayoub and Pietro Mercati are are with the Strategic CAD Lab, Intel Corporation, Hilsboro, OR 97124 USA

Tajana Rosing is with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093 USA

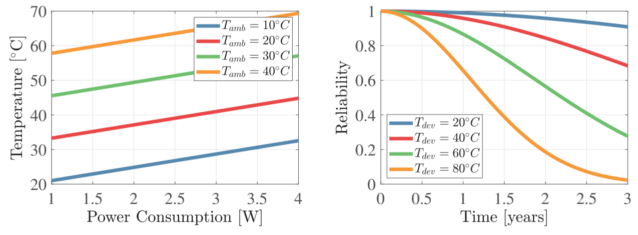


Fig. 1: (a) Device temperature as a function of power dissipation at different ambient temperatures (b) Device reliability over time

An IoT system, as any electronic or mechanical system, is prone to failures. Cisco estimated that for every 100k devices that operate in IoT smart homes, around $6.7M/year are spent for problems related to system failures [3]. The sources of these failures are: user errors, communication problems, power issues, soft and hard errors in hardware. The majority of the errors result in a transient failure and are recoverable without the need of physical human intervention. However, in the case of hard errors, the devices age, degrade, and eventually fail, requiring maintenance for the replacement of defective parts at high costs. In this work, we devote our attention to mitigating *reliability degradation* in IoT devices to defer hard failures.

Reliability degradation of electronic circuits worsens as the technology scales due to intensified effects of various mechanisms such as Time-Dependent Dielectric Breakdown (TDDB), Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) [4], [5], [6]. Degradation is mainly induced by temperature stress, which depends on power dissipated for running workloads and environmental conditions, e.g., ambient temperature. To illustrate this cause-and-effect chain, in Fig. 1a we depict the steady-state temperature of a device as a function of its power dissipation at various ambient temperatures. Also, Fig. 1b shows the reliability over time of the same device as a function of its temperature. The values are based on our measurements in Section VIII. (for temperature) and a reliability model fitted to hypothetical worst-case and best-case temperatures. As observed from the plots, an increase in power dissipation leads to heating of the device, which in turn accelerates reliability degradation.

Recently, due to the shortcomings of traditional cloud-centric IoT (e.g., latency, energy, privacy, cost) [7], [8], Edge Computing [9] is emerging as a promising solution, where data processing is pushed to the edge of the IoT network (as shown in Fig. 2). Since IoT devices at the edge are now capable and powerful enough, Edge Computing envisions to perform data processing and storage on them locally, close

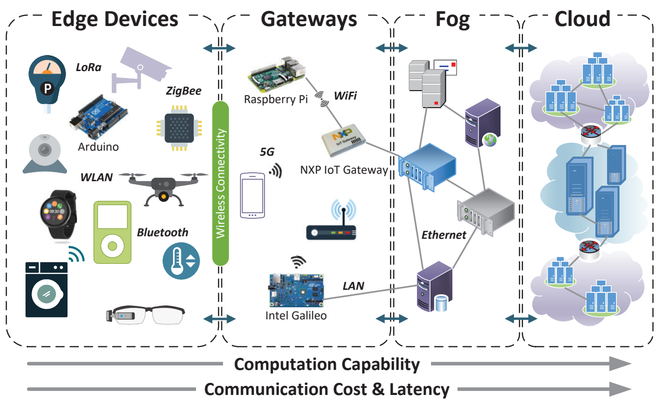


Fig. 2: IoT network architecture

to the source of data. Accordingly, these 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. Fortunately, this stress can be controlled by runtime management techniques to achieve a desired reliability over time [10]. Curbing power dissipation, in particular, helps by lowering the device temperatures and reducing the effect of temperature-driven failure mechanisms [5].

The Edge Computing architecture utilizes gateways to enable application-specific connectivity between edge and fog devices (Fig. 2) [7]. The term “fog” refers to its cloud-like properties, but closer to the “ground”, i.e., closer to the users or the source of data. Being cloud-like is what differentiates fog computing from edge computing; fog devices (e.g., servers) are also in physical proximity to the users, but are still powerful like cloud. The edge refer to low-power IoT devices, or smart objects, mobile phones. As we illustrate in Fig. 2, our definition places edge devices right at the bottom of the network hierarchy and the fog devices very close to the cloud. The gateways have limited computational capabilities compared to fog devices (e.g., high-end servers), but still more capable than low-power sensors, smart objects, and microcontrollers at the edge. A portion of the computation assigned to edge devices can be offloaded to IoT gateways. However, the edge devices cannot independently carry out offloading because the computation resources and communication bandwidth of the gateways are limited, and have to be shared between numerous devices. The offloading amount should be selected in consideration with the Quality of Service (QoS), the energy consumption and reliability of every edge device, and the resources available at the gateway. Several prior works [11], [12], [13] proposed different computation offloading and resource allocation techniques for cooperative operation in the Edge Computing setting, but none considered the reliability of edge devices in their approaches. As edge devices undertake bigger workloads, thermal stress and reliability issues cannot be neglected.

For typical edge computing systems, as studied by prior work [14], [15], [11], [16], improving the energy efficiency of devices while delivering a minimum QoS is the main goal since many edge devices are battery-operated or have limited energy sources [17]. To reduce the number of maintenances performed for battery and component replacement, battery lifetime should be maximized and a certain reliability condition (e.g. minimum MTTF requirement) should be satisfied for the edge device. On the other hand, the level of user’s satisfaction, described by QoS, mostly improves with increased computation. For example, processing data at high sampling rates, making inference from high-resolution data yield better predictions for machine learning tasks, which would improve QoS. A *dynamic* and *scalable* management mechanism is needed to control edge devices such that they satisfy the reliability and QoS requirements in the most energy efficient manner. The necessity for a dynamic solution is due to following reasons: (i) the QoS requirements fluctuate at runtime, (ii) the relative remaining energy of edge devices vary over time, and (iii) the communication bandwidth and the available resources at the gateways can change because of unpredictable environments and other workloads respectively. The edge computing system should quickly adapt to these variations.

In addition to above argumentation, IoT systems usually incorporate many gateways, which provides a degree of freedom to the problem at hand. Edge devices have multiple gateway options to connect and offload computation. It is of great importance to avoid inefficient system operation by properly assigning edge devices to gateways. For example, there may be cases where some gateways are congested with offloaded data despite other gateways being underutilized. This unbalanced employment of gateway resources would lead to suboptimal system performance, thus, the load on the gateways should be distributed evenly. There needs to be a mechanism that intelligently assign edge devices to gateways. Furthermore, offloaded data can be relayed in multiple hops through many edge devices on the path to the gateways. The exact routes from each edge device to their corresponding gateway should also be determined.

Ideally, the gateways must be self-organizing and self-supported [13], with no or minimum dependency on the cloud [7]. In other words, the gateways should handle the management of the system and provide control decisions to the edge devices. This means that a light-weight, low-overhead, dynamic, and scalable solution at the gateways is required for the management to be responsive to dynamic 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 compute-intensive methods on resource-constrained gateways.

In this paper, we present a novel multi-gateway DRM technique for IoT edge devices, taking advantage of the Edge Computing architecture where a portion of the edge devices’ computation 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.

The contributions of this paper are as follows:

- To the best of our knowledge, we are the first to address the reliability management problem in a networked multi-gateway edge computing setting. Unlike the DRM techniques for stand-alone devices, our approach exploits both individual (dynamic voltage and frequency scaling) and network-level (offloading and routing) controls to mitigate reliability degradation.

- We propose a two-level interconnected management scheme, namely the *Intra-Gateway Management* and the *Inter-Gateway Management*, which work together to (i) choose the offloading rates of edge devices, (ii) assign edge devices to gateways, and (iii) decide multi-hop data flow routes and rates in the network.
- For *Intra-Gateway Management*, we formulate a finite horizon nonlinear optimal control problem for finding the best offloading rates for a local network with a single gateway and its associated edge devices. 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.
- For *Inter-Gateway Management*, we construct a routing problem to jointly decide which gateway to offload and which network path to use for communicating data. The solution is linearized and distributed among all edge devices and gateways in the overall network via dual decomposition and subgradient methods.
- Using real measurements to drive trace-driven simulations, we demonstrate that our proposed scheme can achieve a similar battery lifetime and better QoS compared to the state-of-the-art approaches while satisfying reliability requirements, where other approaches fail by a large margin.

II. Related Work

A. Edge Computing in IoT Systems

The IoT contains a large number of battery-powered heterogeneous devices, connected in networks with multiple layers, which should satisfy different service quality requirements in an energy-efficient and reliable manner. Many recent efforts have addressed these challenges in IoT, proposing computation offloading, efficient resource allocation, and QoS management solutions. The definition of QoS in IoT depends on the service it provides, where the service can be described as data acquisition and communication, information processing, or decision making [18]. The majority of previous works dealt with traditional QoS attributes such as service delay and throughput. In [19], the authors present a delay-minimizing collaboration and offloading policy for fog-capable devices that aims to reduce the service delay. They use queueing theory based analytical models to evaluate service delay in IoT edge-fog-cloud architectures and decide on when to offload a task to upper layers. To deal with the uncertainty of task arrivals, a recent study in [20] uses a two-timescale Lyapunov optimization algorithm and makes delay-optimal decisions only based on the system's current state. Such works neglect the other QoS attributes like energy consumption, cost, information accuracy, availability of network resources, etc., which are critical, especially in edge-oriented IoT.

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 [14] and [21] 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 [22], both single-user and multi-user versions of the same problem, in a mobile-edge computing (MEC) setting, are 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 input data of a task among sensor nodes was employed to minimize energy consumption while satisfying a completion time requirement in [15]. Similar problems for partitioning and offloading workloads to fog/cloud were solved by game-theoretic approaches [23], multi-objective optimization [24], heuristic algorithms [16], and primal decomposition [25]. However, the offload target (fog/cloud server) 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. Therefore, the operation of one edge device has an effect on all the other devices in the same network. In [26], the problem of QoS management for IoT edge devices under bandwidth, battery, and processing constraints is addressed. The suggested approach is to partition an application and quantize its input data rate into discrete levels that correspond to different amounts of offloading and QoS. Then, the optimal levels that maximize the overall QoS of the system is computed with dynamic programming. The study in [13] denominates these distinct levels as 'operation modes' and advances the prior work in terms of execution time and memory overhead. Finally, task allocation [27] and task scheduling [12] schemes were proposed to determine where and in which order to execute tasks. In contrast to other works, reference [12] considers the mobility and the ability to perform approximate computing of edge 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 [27], [24], [12], [28]. These problems are commonly formulated as Integer Linear Programming (ILP) problems and solved with heuristics to find the best allocation of application stages/tasks, from a finite set of options, e.g., a few discrete offloading levels. 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, we have a control-theoretic approach; we treat the selection of processing and offloading rates as an optimal control problem.

Our previous work [29] is the first to address the reliability management problem in a networked edge computing system. The problem setting assumes a single-gateway to which edge devices are connected in a star topology, and the proposed solution only controls the offloading rates of edge devices. This paper extends and improves [29] by introducing a two-level management scheme that additionally assigns edge devices to gateways and orchestrates the routing of larger multi-gateway networks, connected in mesh topology. Previously, the management of multi-gateway systems was studied in [30] to improve the service quality of IoT applications under limited network bandwidth. The authors present a trade-based approach in which gateways negotiate and trade edge devices based on battery lifetime and available processing resources. Although the outline of the problem is similar to ours, the specifics of their problem setting and modeling are principally different. They do not consider reliability and solve a multiple knapsack problem over discrete levels for offloading rates, service quality, processing power, and bandwidth.

B. Dynamic Reliability Management

The term *reliability*, especially in networks, is associated with many different types of failures. Almost all of the literature on network reliability focuses on communication link reliability, that is, the situations where the connection between two nodes in the network fails. In some papers, node failures are also included, but they can be mostly categorized into three groups: soft errors (causing random bit flips) [31], software reliability issues [32], or batteries running out of energy [33], [34]. For example, in [32], software failures, message congestion, VM failures on IoT devices are considered, and the failures are modeled as a Poisson process with an average failure rate. There are also some hardware failures discussed in various works (such as [35]), but they consist of superficial models of sensor faults; short faults, constant faults, and noise faults. These types of failures are transient and can be more easily fixed, whereas hard failures are not recoverable. In [36], the authors propose dynamic updates on a reliability function of hard failures, but the failure rate is still modeled as a constant. In comparison, our temperature dependent models, where the failure rates change over time, can capture the dynamic degradation in reliability.

The thermal and reliability aspects of IoT devices are mostly neglected in previous IoT-related work. As IoT devices become more powerful, thermal and reliability issues cannot be ignored and should be taken into consideration in the management strategies. Extensive literature exists for the reliability degradation phenomena on system-on-chips (SoCs). The considered failure mechanisms include TDDB, BTI, and HCI, which all limit device lifetime [10], [37], [6]. 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 [5], [10], [37], [6], task scheduling [38], or both [39]. In [40], 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 [39] implement the above-mentioned mechanisms on a mobile device, showing as much as a one-year improvement on lifetime with dynamic reliability management.

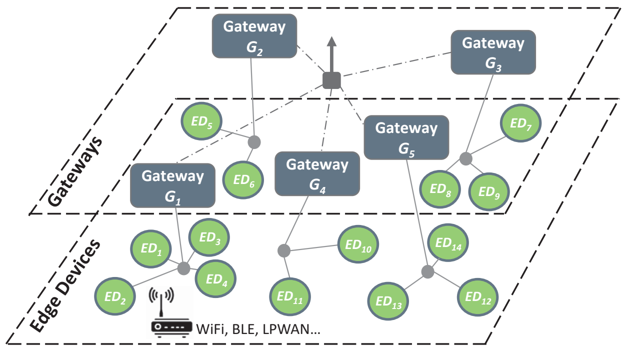


Fig. 3: Multi-gateway IoT network

Despite the impressive results on individual devices, reliability management for networks of IoT devices is an open problem. The recent paper in [17] briefly discussed reliability in the context of IoT and acknowledged that IoT devices can profit from voltage scaling with respect to power and energy. In [29], we showed that the reliability of edge devices can be improved without sacrificing network performance or battery lifetime. To the best of our knowledge, we are the first to propose reliability management for multi-gateway edge computing, which leverages both individual controls (voltage/frequency scaling) and network-level mitigation strategies, such as computation offloading and routing.

In summary, none of the of the related works is applicable to our problem because they either (i) neglect reliability, or some QoS attributes such as energy consumption, availability of network resources, which are critical in edge-oriented IoT, (ii) assume the offload target (fog/cloud server) to be very powerful and fast, or to have unlimited resources, (iii) consider one edge/mobile device, without accounting for resource contention between the network devices, (iv) formulate the problem as task allocation with a few discrete offloading levels, (v) study only single-gateway IoT systems.

III. System Model

The envisioned IoT network architecture has multiple layers comprising edge devices, gateways, fog, and cloud servers as illustrated in Fig. 2. The IoT edge devices sense information from physical phenomena and send preprocessed data to a gateway node, which aggregates the streams of sensed data in real time, processes, and sends them to the central servers, e.g., fog, cloudlets, or cloud servers for storage or further analysis. For the edge computing setting, we focus on the management in the first two layers: the edge and the gateway layer.

A. Network Architecture

We consider an IoT network composed of N edge devices $\boldsymbol{ED} = \{ED_1, ED_2, ..., ED_N\}$ and M gateways $\boldsymbol{G} =$

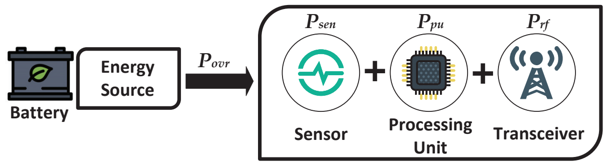


Fig. 4: IoT device model

$\{G_1, G_2, ..., G_M\}$. Each gateway G_j has a subset of $N_j < N$ associated edge devices, which together form a *local network* as shown in Fig. 3. We denote O_j as the set of edge devices connected to gateway j, with cardinality $|O_j| = N_j$. The notation $ED_i \in O_j$ implies that edge device i is in the local network of G_j. It should be noted that this association is not permanent; the edge devices are assumed to be able to dynamically change the gateway to which they are connected. The gateway can either directly relay the processed data from the edge devices to upper network layers, or it can help with computation and process a portion of the raw data offloaded from the edge devices.

In the local network, edge devices share the limited resources of gateway's computation power and communication bandwidth. They communicate with WiFi (IEEE 802.11) or low-power, low-bandwidth wireless technologies such as BLE (Bluetooth Low Energy), ZigBee (IEEE 802.15.4), and LPWAN (Low-Power Wide-Area Network). The bandwidth BW_j is the total available bandwidth of the local network associated with gateway G_j, where the wireless medium is shared between the edge devices and the gateway. It is assumed to be varying because of the possible changes in the communication medium and interference from external sources. We assume a mesh topology within a network, where connection is allowed between every edge device depending on the maximum distance they can transmit. Let S_i denote the set of neighboring devices to which node i can send packets to. Then, $S_i = \{j : d_{i,j} < d_{max}\}$, where $d_{i,j}$ is the distance between devices i and j and d_{max} is the distance of transmission with maximum power. The notation $j \in S_i$ is used to show that j is a neighbor of i and they can communicate. The devices can have mobility, in which case the neighbors change depending on the locations of the devices. The location of all devices are assumed to be known, either by GPS or other localization methods.

B. Device Models

As depicted in Fig. 4, each IoT device is equipped with: (i) sensors, (ii) a processing unit, (iii) a transceiver, and (iv) an energy source. The sensors sense physical phenomena and sample input data, the processing unit (e.g. CPU, GPU, FPGA) performs computation, and the transceiver carries out the communication between the edge devices and the gateway. We assume that both the edge devices and the gateway abide by similar device models but with different parameters. The main distinction between them is edge devices being more resource-constrained, that is, lower communication, storage, and computation capabilities. In the following, we describe the power, temperature, reliability, and battery models of the devices.

Power Model. The overall power consumption P_{ovr} of the edge device includes the sensing power P_{sen} of the sensors, the computation power P_{pu} of the processing unit, and the communication power P_{rf} of the transceiver.

$$P_{ovr} = P_{sen} + P_{pu} + P_{rf} \tag{1}$$

The power consumption P_{pu} of a processing unit can be modeled through Equation (2) as the sum two contributions: leakage power P_{lea} (also called as static power) and dynamic power P_{dyn}. The dynamic power is resulted from the logic gate switching and is dependent on the operating frequency f. The leakage power is affected by temperature T and it can account as much as 50% of the total power consumption in current CMOS technologies [41].

$$P_{pu} = P_{dyn} + P_{lea} = \alpha C^{eff} V_{dd}^2 f + V_{dd}(b_T T^2 e^{\frac{k}{T}} + I_{gate}) \tag{2}$$

Here, α and C^{eff} are the activity factor and the effective switching capacitance. The coefficient b_T is a technology dependent constant, k is the Boltzmann constant, and I_{gate} is the gate leakage current which can be assumed constant. Since the clock frequency f depends linearly on voltage V_{dd} [42], a simplified model that accounts for both dynamic and leakage power can be given as $P_{pu} = af^3 + bf$.

The communication power consumption is determined by the rate of the bits transmitted over the wireless channel. The energy consumption of a IEEE 802.11n or IEEE 802.15.4 wireless node is dominated by the transmit or receive modes, and their costs are approximately the same. The communication cost is characterized by the empirical transmission power model [43] and the required power P^{rf} to transmit L bits/second is governed by:

$$P_{rf} = \rho_1(d)\frac{L}{g} + \rho_2 \tag{3}$$

where $\rho_1(d) \geq 0$ denotes the energy coefficient monotonically increasing in distance d; the most common such function is $\rho_1(d) = C_f + C_s d^\beta$ where C_f,C_s are given constants depending on channel attenuation as well as specific modulation techniques and β is a constant dependent on the medium. g denotes channel state and ρ_2 is the static power consumed by RF circuits. Finally, the sensing power consumption can be simply modeled as a linear function of the sampling rate of the sensor.

$$P_{sen} = c_s \lambda \tag{4}$$

where λ is the sampling rate, or the output traffic rate of the sensor.

Battery Model. Not just the net amount, but the way in which the power is consumed, that is, the current-extraction patterns and the employed current levels play a significant role in battery depletion [44]. Therefore, it is inaccurate to assume linear energy depletion with respect to the power consumed/current drawn, a dynamic battery model is needed to realistically capture the influence of power consumption on the battery. We use *Temperature Dependent Kinetic Battery Model* (T-KiBaM) [45], a dynamic model which can describe the nonlinear characteristics of available battery capacity. It is able to accurately characterize the two important effects (rate capacity effect, and recovery effect) that make battery performance nonlinear [44]. The effective capacity of a battery drops for higher discharge rates. This effect is termed as *rate capacity effect*. If there are idle periods in discharging, the battery can partially recover the capacity lost in previous discharge periods. This effect is known as *recovery effect*. It was shown in [46] that using battery models that capture these effects results in more accurate optimization and control algorithms, and hence better network management techniques.

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 the secondary charge flowing towards the ACT. In this way, T-KiBaM successfully models the recovery and rate capacity effects. The flow rate between the two tanks is regulated by their height difference and the temperature. The battery is denoted empty when its ACT depletes. Let $P_{ovr} = P_{pu} + P_{rf} + P_{sen}$ 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. Then, Equation (5) gives the system of differential equations that describes T-KiBaM. At any time instant, $q_A + q_B$ is the total available charge in the battery. Parameters κ and c are predefined constants that can be obtained using the battery data-sheets or through experimental measurements [45].

$$\begin{cases} \dfrac{dq_A}{dt} = -\kappa(1-c)q_A + (\kappa c)q_B - \dfrac{P_{ovr}}{V_{dd}} \\ \dfrac{dq_B}{dt} = \kappa(1-c)q_A - (\kappa c)q_B \end{cases} \tag{5}$$

Temperature Model. Temperature of a device depends on the power dissipated and ambient temperature. We define the power consumption vector of the edge device, $P_{ed} = [P_{pu}, P_{rf}]^T$, only including the computation and communication terms. Accordingly, let the heat sources be the PU and RF and let $T_{ed}(k)$ represent the vector of temperatures observed by thermal sensors at time instant k. The heat sources are assumed to have one thermal sensor measuring its temperature. Then, temperature $T_{ed}(k+1)$ at time instant $k+1$ can be predicted given the current temperature $T_{ed}(k)$ and power $P_{ed}(k)$ at time k. The discrete-time state-space model of the device's thermal behavior is expressed in the following equation [47].

$$T_{ed}(k+1) = A_T \cdot T_{ed}(k) + B_T \cdot P_{ed}(k) + C_T \cdot T_{amb}(k) \tag{6}$$

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. Deriving the model (i.e. matrices A,B,C) of Equation (6) by only accessing power and temperature is a blind identification problem. To solve this problem, we use a numerical algorithm for subspace system identification (N4SID [48]) and derive the model from measured power and temperature traces.

Reliability Model. The main degradation mechanisms affecting integrated circuits are Time Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI), Electromigration (EM), and Thermal Cycling (TC) [10]. Models have been developed for MTTF for each degradation phenomenon, which show a strong (exponential) dependence on temperature. For example, the MTTF for TDDB is described by Equation (7).

$$MTTF_{TDDB} = A_0 \exp - \gamma E_{ox} \exp \frac{E_a}{k_B T} \tag{7}$$

A_0 is a constant determined empirically, E_{ox} is the electric field across the dielectric, γ is the field acceleration parameter, E_a is the activation energy, and k_B is the Boltzmann constant. The MTTF for NBTI is:

$$MTTF_{NBTI} = A_0 (\frac{1}{V})^{\gamma_v} \exp \frac{E_a}{k_B T} \tag{8}$$

where γ_v is the voltage acceleration factor and V is the applied voltage. The MTTF for HCI is described by the Eyting model, expressed in Equation (9) for N-channel devices.

$$MTTF_{HCI} = B I_{sub}^{-C_{mat}} \exp \frac{E_a}{k_B T} \tag{9}$$

Here, I_{sub} is the peak substrate current during stressing, C_{mat} is a material dependent constant and B is a scale factor, function of technological parameters.

Similar to power and temperature models, for reliability models we divide the device into structures – PU & RF – and apply the analytic models to each structure as an aggregate. To obtain the overall MTTF of an edge device, we combine the effects of different failure mechanisms, across these different structures. A standard model used by the industry is the sum-of-failure-rates (SOFR) model [10], which makes the assumption that the device is a series failure system, in other words, the first instance of any structure failing due to any failure mechanism causes the entire device to fail. Hence:

$$MTTF_{ed} = \frac{1}{\sum_{i=1}^{n_s} \sum_{j=1}^{n_m} \frac{1}{MTTF_{ij}}} \tag{10}$$

where $MTTF_{ij}$ is the MTTF of the i^{th} structure due to the j^{th} failure mechanism. The variables n_s and n_m are the number of structures and mechanisms, respectively.

MTTF for each degradation mechanism is related to a reliability function as expressed by Equation (11), where reliability $R(t)$ is the function depicting the probability of not having failures before a given time t, defined in the interval $[0, 1]$. Compared to MTTF, reliability is a function of time, so it is more suited for the purpose of dynamic management [39].

$$MTTF = \int_0^\infty R(t)dt \tag{11}$$

The reliability function $R(t)$, in general, is expressed as a monotonically decreasing exponential function of time and temperature [5].

$$R(t) = \gamma_1 \exp(-\frac{E_a}{k_B T}) \exp(-\gamma_2 t) \tag{12}$$

where γ_1, γ_2 are the constants depending on the respective mechanism. The expression in Equation (12) is only representative of static systems because it assumes a constant temperature applied from time $t = 0$. The workloads and

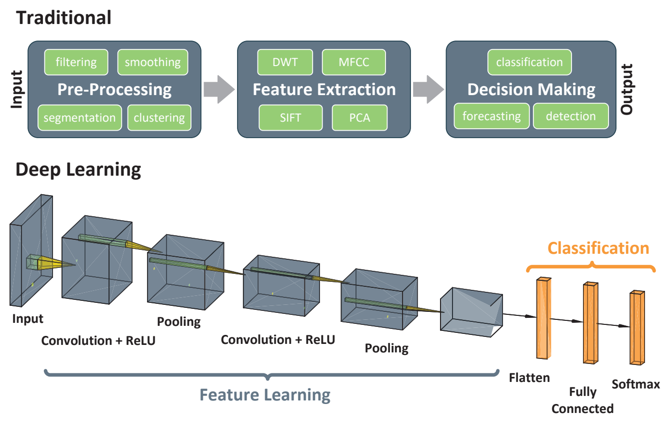


Fig. 5: Segmented machine learning applications

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:

$$\Delta R(t_{eqv,1}, T_2) = \Delta R(t_1, T_1) \tag{13}$$

The equivalent degradation time $t_{eqv,1}$ can be computed using Equation (12). 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.

$$\begin{aligned}\Delta R(t_{eqv,k-1}, T_{k-1,k}) &= R_0 - R_d(k-1) = \Delta R|_{t=t_{k-1}}\\ R_d(k) &= R(t_{eqv,k-1} + t_{k-1,k}, T_{k-1,k})\end{aligned} \tag{14}$$

In Equation (14), 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 $t = 0$.

Similar to the system MTTF expression in Equation (10), multiple reliability functions can be combined into a single one when considering the effect of multiple mechanisms and structures together as a series failure system.

$$R_{d,ed}(k) = \prod_{i=1}^{n_s} \prod_{j=1}^{n_m} R_{d,ij}(k) \tag{15}$$

The variables n_s and n_m are the number of structures and failure mechanisms, respectively. $R_{d,ij}$ is the reliability of the i^{th} structure modeled by the j^{th} failure mechanism.

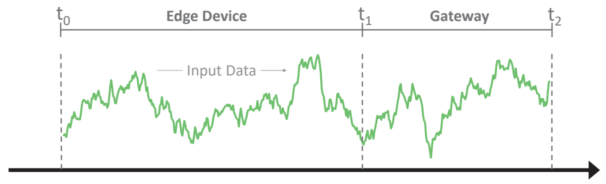


Fig. 6: Data partitioning and offloading over time

C. Application Model

We consider the cooperative computing setting in which edge devices can execute applications with the help of gateways. In the following, we elaborate the application model and describe the operation of the edge devices and the gateways.

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 [9]. As illustrated in Fig. 5, traditional machine learning (ML) approaches and deep neural networks (DNN) are examples of commonly used applications in IoT systems that can be segmented and mapped to different IoT devices. Several works considering general ML applications [13], [49], [50] and DNNs [51], [52] exist, though, the segmentation of applications and the distribution process are beyond the scope of this paper. In our work, we assume that this segmentation and distribution process is governed by an external management mechanism, such as [27]. Therefore, the edge devices in our network are dynamically being assigned different tasks.

The tasks can be executed either locally at the edge devices or remotely on the gateways via computation offloading. In particular, the *input data* of the tasks can be partitioned and offloaded (communicated) to the gateways, as illustrated in Fig. 6. 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. As a concrete example, let us consider a system that runs a feature extraction algorithm. The application code is assumed to be already present on both devices. Therefore, the features can be extracted from “raw” sensor data at the edge devices, then the processed features are communicated to the gateway. Another option is to send the raw data directly to the gateway and extract the features there. Input data partition comes into play at this stage. For example, the sensor of the edge may be device generating 10kB of data every 5 seconds, i.e., at a rate of 2kB/s. If it sends the first 6kB chunk of this data to the gateway and process (extract features) the next 4kB locally at the edge devices, then the offloading rate and local processing rates are 1.2kB/s and 0.8kB/s, respectively.

It is worth noting that sometimes an application (e.g. geo-distributed MapReduce [53]) can be breakable into tasks which do not exhibit dependencies across partitions of its input [15]. Provided this condition, the edge device and the gateways can also be assumed to be able to run different tasks. 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. According to the delay requirements of the

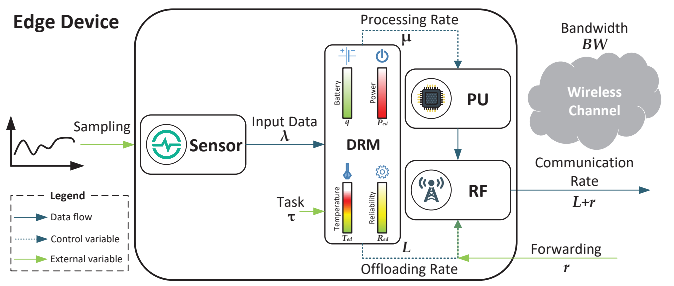


Fig. 7: Structure of the edge device

application, the tasks can be categorized into *delay-sensitive* and *delay-tolerant* (i.e., best-effort) ones [18]. The delay-sensitive tasks are required to be served in a timely fashion, and have hard deadline constraints usually from milliseconds to tens of milliseconds. In contrast, delay-tolerant tasks, such as data-based applications as in personal health analytics and ML model training are tolerant to certain delays. Hence, we consider soft deadlines for delay-tolerant tasks and hard deadlines for delay-sensitive tasks.

D. Network Operation

IoT traffic can be roughly categorized into periodic and event-based modes of communication [54]. Some applications will always be event-driven, but still periodicity can ensue. For example, motion detection sensors in smart homes activate roughly at the same time every day, when leaving for work and returning home, in a predictable, periodic manner. In addition, many IoT devices from other fields of application such as smart grids, environmental monitoring etc. often intrinsically generate and communicate data in a periodic fashion. In our work, we assume that the input traffic generated by sensors of ED_i is periodic with a period T_i and a deterministic arrival rate λ_i. Depending on the tasks and QoS requirements, the data arrival rate can differ.

The operation of an edge device is illustrated in Fig. 7, with *local data processing* at its processing unit (PU) and network communication for *data offloading* and *data forwarding* at its transceiver (RF). The rate at which the input traffic is routed to the gateways through RF is L_i, denoted as the offloading rate. There is also incoming external data from other edge devices to be relayed, since mesh network topology is assumed. We use r_i to denote the total forwarding 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 (related by its IPC_m). 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 devices is communicated to the gateways wirelessly. Each edge device is assigned to a single gateway and all of its data should be forwarded to only that gateway. However, since the network topology is mesh, devices can cooperate to distribute and relay data in a multi-hop fashion. Our proposed inter-gateway management framework chooses the target gateways and data forwarding routes for

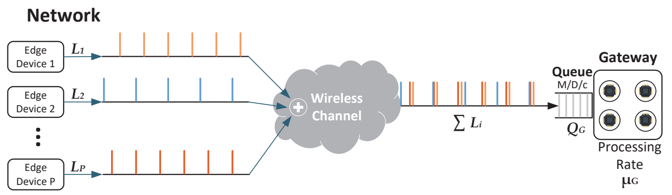


Fig. 8: Local network operation

every edge device in the network. The gateways receive the superposition of offloaded periodic traffics from a number of unsynchronized edge devices (Fig. 8). According to the Palm-Khintchine theorem, this aggregated traffic for each gateway can be approximated with a Poisson process with the arrival rate $\sum_{ED_i \in O_j} L_i$, that is, the sum of offloading rates of the associated edge devices [54]. The computing resources of a gateway is adequate for processing data for several tasks from multiple edge devices simultaneously. We assume that there are c homogeneous computation cores in a gateway's SoC, working with a deterministic processing rate $\mu_{G,j}$. Also, unlike the edge devices, memory resources of the gateways are sufficient to be able to hold a queue of incoming data. Therefore, the gateways employ a queueing structure of type M/D/c [55], denoted $Q_{G,j}$. The discrete queue dynamics at the input of the gateways are as follows:

$$Q_{G,j}(k+1) = [Q_{G,j}(k) + \sum L_i(k) - \mu_{G,j}(k)]^+ \qquad (16)$$

where $Q_{G,j}(k)$ denotes the queue length of gateway j at time instant k, in bits, and $[x]^+ = max(x, 0)$. $\mu_{G,j}$ is the total computation resources available at the gateway, in bits per unit time. We assume that the amount of $\mu_{G,j}$ can dynamically change depending on the overall network operation and we do not have control over it.

For delay-tolerant tasks, it is enough to finitely maintain the queue lengths in Equation (16). This assures that all arrived tasks are served within finite time. However, for delay-sensitive tasks, we need to provide a delay guarantee. We introduce a delay aware virtual queue based on the ϵ-persistent queue technique [56] to ensure that the tasks are finished with a delay lower than D_m.

Delay-Aware Virtual Queue. In order to guarantee the maximum delay $D_{m,i}$ for task m associated with edge device i, offloaded to gateway j, we employ a delay-aware virtual queue $Z_{G,j}$ whose equation is shown below:

$$Z_{G,j}(k+1) = \begin{cases} 0, \text{ when } Q_{G,j}(k) \leq \mu_{G,j}(k) \\ [Z_{G,j}(k) - \mu_{G,j}(k) + \epsilon_{G,j}]^+, \text{ o.w} \end{cases}$$

where $\epsilon_{G,j}$ is a pre-specified constant based on the delay constraint. $Z_{G,j}(k)$ has the same service process as $Q_{G,j}(k)$ but has an additional constant arriving process $\epsilon_{G,j}$ whenever the actual queue backlog $Q_G(k)$ is larger than $\mu_{G,j}(k)$. This ensures that the virtual queue grows only when there exists data in the original queue that have not been served. Therefore, if there is data from a task staying in the waiting queue for a long time, the queue length of $Z_{G,j}(k)$ will continue to grow. Any algorithm that maintains bounded $Z_{G,j}(k)$ and $Q_{G,j}(k)$ values also ensures persistent service with bounded worst-case delay. This maximum delay can be expressed in terms of the maximum queue lengths $Q_{G,j}^{max}$ and $Z_{G,j}^{max}$. For a time slot k, if the system can be controlled to ensure that $Q_{G,j}(k) < Q_{G,j}^{max}$ and $Z_{G,j}(k) < Z_{G,j}^{max}$, then any task is fulfilled with a maximum delay W^{max} defined as follows:

$$W^{max} = \lceil (Q_{G,j}^{max} + Z_{G,j}^{max})/\epsilon_{G,j} \rceil \tag{17}$$

Given the above property, we can choose the appropriate $\epsilon_{G,j}$ for each task to ensure that it can not exceed its maximum delay $D_{m,i}$ (i.e. $W^{max} < D_{m,i}$). The original queue Q_G exists in the form of a buffer structure in the system. The received data packets wait in this buffer until they can be served by the gateway. On the other hand, the virtual queue dynamics are implemented by the tracking the original queue and increasing/decreasing the virtual length accordingly.

To summarize, the edge devices produce input traffic via sensors, run different tasks, and process data. As a result of on-board computation, they dissipate a certain power, consume battery energy, heat up, degrade, and hence lose reliability. We provide all the associated device and application models. The edge devices can be connected and offload computation to any of the gateways in the network. The operation of one edge device has an effect on all other devices in the same network, which is formulated by the queueing model. We use the described system model in our problem formulation.

IV. Problem Formulation

In the following, we formalize our problem based on the network and device models presented. The goal of this section is to express the problem in a mathematical framework and relate it to a family of problems from optimization and control fields. We next provide the methods and the tools to solve it in Section V. Table I provides the list of symbols that are used in problem formulation, in the order of appearance throughout the paper.

The target for the above-mentioned multi-gateway system is to have an energy-efficient and reliable operation without sacrificing performance. To achieve this objective, we define three interdependent problems:

(i) choosing the data offloading rates of edge devices,
(ii) assigning edge devices to gateways, and
(iii) deciding multi-hop data flow routes and rates in the network.

We treat problem (i) individually whereas problems (ii) and (iii) are combined. The reason for this particular choice of partitioning is clarified in Section V.

First, we formulate the problem of finding the optimal offloading rates for a local network with a gateway and its associated edge devices. This is called the *Intra-Gateway Problem* since it can be solved by single gateway and the solution depends only on the local network. Then, considering the complete multi-gateway network with all edge devices, we construct a routing problem to jointly decide which gateway to offload and which network path to use for communicating data. This is called the *Inter-Gateway Problem* as it requires global effort from all the devices in the complete network covering multiple gateways.

TABLE I: Nomenclature

Symbol	Definition
ED	Edge device
G	Gateway
N	Number of edge devices
M	Number of gateways
S_i	Set of neighboring devices to node i
BW	Local network bandwidth
P_{ovr}	Overall power consumption of an edge device
f	Device operating frequency
q_A	Total charge in battery Available Charge Tank
q_B	Total charge in battery Bound Charge Tank
T_{ed}	Vector of temperatures of an edge device
T_{amb}	Ambient temperature
$MTTF_{ed}$	Mean time to failure of an edge device
$R_{d,ed}$	Dynamic reliability of an edge device
λ	Input traffic data rate i
L	Data offloading rate
r	Data forwarding rate
μ	Data processing rate
μ_G	Gateway processing rate
Q_G	Gateway queue length
Z_G	Gateway virtual queue length

A. Intra-Gateway Problem

The gateways $\boldsymbol{G}$ are only responsible for the edge devices $\boldsymbol{ED}$ in their own local network, i.e, if $ED_i \in O_j$. Therefore, the *Intra-Gateway Problem* can be formulated separately for each local network. The goal is to maximize the remaining energy in the batteries of edge devices under QoS and reliability constraints. We assume that the gateway can have its energy supplied by the grid and reliability is less of a concern due to available preventative measures (i.e., access to cooling and effortless maintenance).

Cost Function: The cost function of the control problem is the sum of battery energies of all edge devices in the local network. We define the following objective for finite horizon optimal control of j-th local network:

$$\min_{(\boldsymbol{\mu},\mathbf{L})} \sum_{k=0}^{T_f-1} -\|\mathbf{1}^T\overline{\mathbf{q}}(k)\|^2 = \sum_{k=0}^{T_f-1}\sum_{i=1}^{N_j} -\|\mathbf{1}^T\mathbf{q}_i(k)\|^2 \tag{18}$$

where $(\boldsymbol{\mu},\mathbf{L}) \triangleq (\mu_1(k),...,\mu_P(k),L_1(k),...,L_P(k))_{k=0}^{T_f-1}$. The vector $\mathbf{q}_i(k) = [q_{i,A}(k), q_{i,B}(k)]^T$ is the battery charge vector and $\overline{\mathbf{q}}$ denotes the combined vector of all edge devices.

Constraints: There are three QoS requirements that should be satisfied at any time instant k and a terminal reliability constraint that should be satisfied at the final time instant T_f:

1) The maximum task delay $D_{m,i}$ should be met for every edge device i and task m. Then, the delay experienced at the gateway queue should be less than $D_{m,i}$, which is ensured if the length of gateway queue is smaller than a value $Q_{G,j}^{max}$, i.e., $Q_{G,j}(k) < Q_{G,j}^{max}$.
2) Bandwidth utilization should not exceed BW_j. The bandwidth utilization of an edge device i is L_i, hence the corresponding constraint is $\sum_{i=1}^{P} L_i(k) \leq BW_j$.
3) Depending on the application, there is a certain data arrival and service rate at each edge device determined by QoS requirements. We define this *target* rate as λ_i^{target}. The sum of data processed locally and offloaded should be equal to the *target*, i.e., $\mu_i(k) + L_i(k) = \lambda_i^{target}(k)$.

4) The dynamic reliability $R^{sys}_{d,i}$ (Equation 15) of each device at the end of the horizon should be at least the target reliability R_{target}, i.e, $R^{sys}_{d,i}(T_f) \geq R^{target}$

Control Variables: The two performance-related state variables to be controlled for each edge device are: (i) PU processing rate $\mu_i(f)$ and (ii) RF communication (offloading) rate L_i. Then, the control variables include the required change in the operating frequency Δf_i and change in the communication rate ΔL_i.

All in all, we define the following discrete-time finite horizon optimal control problem:

$$\min_{\Delta\mathbf{f},\Delta\mathbf{L}} \sum_{k=0}^{T_f-1} \sum_{i=1}^{N_j} -\|\mathbf{1}^T \mathbf{q}_i(k)\|^2 \tag{19}$$

$$\begin{aligned}
&\text{s.t.} \quad i = 1, ..., N_j \quad k = 0, ..., T_f-1\\
&q_{A,i}(k+1) = A_q q_{A,i}(k) + B_q q_{B,i}(k) - \frac{P_{ovr,i}(k)}{V_{dd}}\\
&q_{B,i}(k+1) = C_q q_{A,i}(k) + D_q q_{B,i}(k)\\
&P_{ovr,i}(k) = a_i f_i^3(k) + b_i f_i(k) + c_{s,i}\lambda_i(k) + \rho_1 \frac{L_i(k)+r_i(k)}{g} + \rho_2\\
&T_{ed,i}(k+1) = A_T T_{ed,i}(k) + B_T P_{ed,i}(k) + C_T T_{amb,i}(k)\\
&Q_{G,j}(k+1) = [Q_{G,j}(k) + \textstyle\sum_{i=1}^{P} L_i(k) - \mu_{G,j}(k)]^+\\
&Z_{G,j}(k+1) = [Z_{G,j}(k) - \mu_{G,j}(k) + \epsilon_{G,j}]^+\\
&\mu_i(k+1) = \mu_i(k) + d_i \Delta f_i\\
&L_i(k+1) = L_i(k) + \Delta L_i\\
&Q_{G,j}(k) < Q^{max}_{G,j}\\
&Z_{G,j}(k) < Z^{max}_{G,j}\\
&\textstyle\sum_{i=1}^{P} L_i(k) \leq BW_j\\
&\mu_i(k) + L_i(k) = \lambda_i^{target}(k)\\
&R_{d,i}(T_f) \geq R^{target}\\
&|\Delta f_i| \leq \Delta f^{max}, \quad |\Delta L_i| \leq \Delta L^{max}
\end{aligned}$$

where we discretized the battery dynamic equations from Equation (5) with state variables $q_{A,i}$ and $q_{B,i}$, representing the charge level of edge device i at time instant k. Overall power consumption $P_{ovr,i}$ is expressed in terms of processing, communication, and sensing rates. On the other hand, $P_{ed,i} = [P_{pu,i}, P_{rf,i}]^T$, a vector of PU and RF power consumption, is used in the temperature dynamics equation to compute $T_{ed,i} = [T_{pu,i}, T_{rf,i}]^T$. We define μ_i and L_i as state variables which are controlled by the inputs Δf_i and ΔL_i: respectively the change in the operating frequency and the change in the offloading rate. By doing this and imposing magnitude constraints on the new control variables, we ensure a smooth transition in both processing and offloading rates.

B. *Inter-Gateway Problem*

The solution to the *Intra-Gateway Problem* finds the offloading rates for every edge device, but it does not specify how the offloaded data should be communicated to gateways. As the network is assumed to have mesh topology, data can be forwarded in multiple hops through many edge devices on the path. The exact routes from each edge device to the gateways should be determined. Also, the *Intra-Gateway Problem* is formulated for a fixed set of edge devices in the local networks. However, as stated in Section III.D, edge devices have multiple choices for which gateway to offload data. These choices should be made considering the state of the system. We construct the *Inter-Gateway Problem* whose solution gives the edge device to gateway assignments, as well as the routing between them.

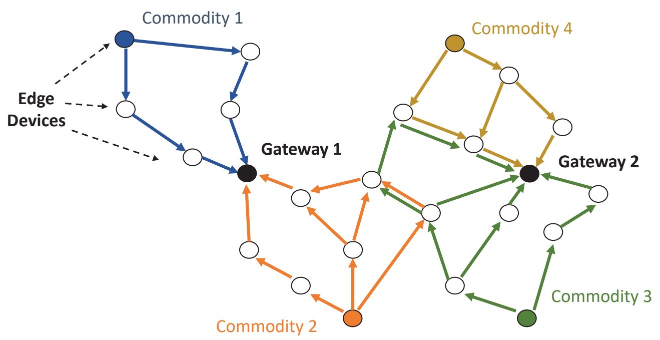


Fig. 9: Multicommodity flow multiple sink routing

The desired joint problem can be composed into a single network routing problem with multicommodity flows and multiple sinks. The goal is to find the maximum lifetime routing. From the *Intra-Gateway Problem*'s solution, we obtain L_i, the rate at which data is generated at edge device i. We consider the data from different edge devices as different *commodities*. This data needs to be communicated to any of the gateways in the network, resulting in the multicommodity flow multiple sink routing problem. We assume that in general, each commodity should only be communicated to a single gateway, that is, data from one edge device cannot be distributed to multiple gateways. For example, if the data is sequential (e.g., time-series data), then it should be received at one gateway in the same order to be processed correctly. If the data is distributed to many gateways and not received as a whole at a single gateway, the task cannot be carried out. The packet transmission is thus unicast. An alternative solution for when this assumption does not hold is discussed in Section VII-A. An example solution for our problem setting is illustrated in Fig. 9 for a network with two gateways.

For notational convenience in the routing problem, consider the network nodes numbered from *1* to *N* denote the edge devices and *N+1* to *N+M* denote the gateways. In other words, $i \in V_{ED}$ and $j \in V_G$ for the edge devices and gateways respectively, where $V_{ED} = \{1, ..., N\}$ and $V_G = \{N+1, ..., N+M\}$. Let r^i_{kl} denote the rate of data flow from edge device k to any node $l \in S_k$, carrying edge device i's commodity. The aggregate data rate for the unidirectional link from edge device k to l is denoted by r_{kl} and is equal to $\sum_{i=1}^{N} r^i_{kl}$. For simplicity of notation, we stack up all r_{kl} into a single vector and denote network flow as $\mathbf{r} = \{r^i_{kl}\}$, Then, the lifetime of edge device i under flow $\mathbf{r}$ is given by

$$MTTF_{ed,i} = \gamma_c \text{exp} \frac{E_a}{k_B T_{ed,i}(\mathbf{r})} \tag{20}$$

We define lifetime in terms of mean time to failure, where Equation (20) is a generalized form of MTTF definitions in

Equations (7), (8), (9) and coefficient γ_c encompasses the multiplicative terms in the respective formulas. Temperature $T(\mathbf{r})$ is a function of network flow as it alters according to device power dissipation (Equation (6)), which in turn relates to data flow through Equation (3).

We assume that a network fails with the first node's failure as a common definition. This definition is one of the most prevalent in literature [33] and was used in many recent works [57], [58]. In this case, network MTTF under flow $\mathbf{r}$ is the minimum of any node in the network, i.e.

$$MTTF_{net}(\mathbf{r}) = \min_{i \in N} MTTF_{ed,i}(\mathbf{r}) \tag{21}$$

Our goal is to find a solution for the flow $\mathbf{r} = \{r^i_{kl}\}$ that maximizes the network lifetime. Hence, we formulate the following problem.

$$\begin{aligned}
\underset{\mathbf{r},\mathbf{X}_{assign}}{\text{maximize}} \quad & \min_{i \in V_{ED}} MTTF_{ed,i}(\mathbf{r}) && (22)\\
\text{subject to} \quad & \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = L_i, \; \forall i,k \in V_{ED}, \; i = k \\
& \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = 0, \; \forall i,k \in V_{ED}, \; i \neq k \\
& r^i_{kl} \geq 0, \; \forall i,k \in V_{ED}, \; \forall l \in S_k \\
& \sum_{k \in V_{ED}} \quad r^i_{kl} = L_i, \; \forall i \in V_{ED}, \; \forall \{l \in V_G \mid x_{il} = 1\} \\
& \sum_{j=1}^{M} x_{ij} = 1, \; \forall i \in \{1, ..., N\} \\
& x_{ij} \in \{0,1\}, \; \forall i \in \{1,...,N\}, \forall j \in \{1,...,M\}
\end{aligned}$$

The optimization variables are r^i_{kl} and x_{ij}. $\mathbf{X}_{assign}$ is the assignment matrix in which elements x_{ij} assume value 1 if edge device i is assigned to gateway j and 0 otherwise. The matrix $\mathbf{X}_{assign} \in \mathbb{R}^{NxM}$ has only one element equal to 1 for each row. This is because data from one edge device cannot be distributed to multiple gateways so each commodity should only be communicating to a single gateway. The first two constraints are the flow conservation equations at each node. The difference between incoming and outgoing flows for each commodity is equal to the data generation rate. We express the condition on commodities that restrict them to be communicated to a single gateway by the fourth constraint. The summation of all outgoing flows towards the l-th gateway for the i-th commodity should be L_i.

V. Proposed Approach: Overview

In this section, we first present the general solution framework and briefly describe its operation. Subsequently, we break down and analyze the proposed solution in further detail. Fig. 10 depicts an overview of the proposed two-level management scheme. The overall management methodology is an interplay between *Intra-Gateway Management* and *Inter-Gateway Management* components:

- *Intra-Gateway Management* is responsible for choosing the local processing and offloading rates of edge devices. Each gateway runs it separately for the edge devices in their own local networks.

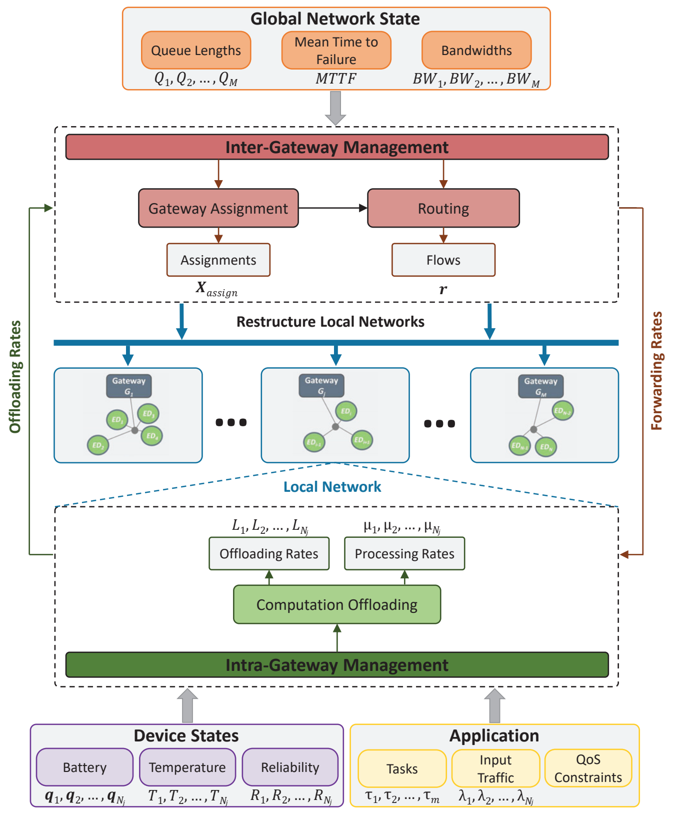


Fig. 10: Overall architecture of the proposed management scheme

- *Inter-Gateway Management* assigns edge devices to gateways and decides multi-hop data flow routes and rates in the network. It is carried out with collaborative effort from all devices.

The two components work together in a cyclical fashion; one computes its solution based on the other's output. *Inter-Gateway Management* takes as input the data offloading rates set by *Intra-Gateway Management*. On the other hand, *Intra-Gateway Management* determines optimal offloading rates in accordance with the gateway assignments and the data forwarding rates of edge devices.

At the beginning of system operation, the gateways are evenly matched with the closest edge devices and they establish single-hop connections. M disconnected local networks are formed with an average of N/M edge devices per gateway. Based on the initial assignments, optimal offloading rates for edge devices are calculated via *Intra-Gateway Management* separately at each local network. *Inter-Gateway Management* then uses these offloading rates to make gateway assignment and routing decisions. It assigns edge devices to gateways primarily based on fairness such that each gateway receives similar amounts of offloaded data. At this stage, the initial topology of the network is changed and edge devices have new gateway pairs. The topology is not restricted to single-hop connections, so data can be forwarded in multiple hops through many edge devices on the path. *Inter-Gateway Management* lastly adjusts communication paths and data flow rates on the communications links.

After the initialization phase, both management components continue to work in tandem. *Inter-Gateway Management*'s routing introduces additional communication load to some edge devices due to multi-hop communication, which was not assumed at the system start. *Intra-Gateway Management* accordingly adjusts processing and offloading rates to compensate for the additional data forwarding load on the edge devices. The edge computing system is already dynamic due to variable workloads and resources, fluctuating temperatures, etc., so the solution is continuously updated at certain intervals.

The gateway assignments, data flow rates and paths are also not fixed. *Inter-Gateway Management* updates the solution under the following conditions:

1) Periodically, at regular intervals,
2) If the bandwidth allocation of any local network is over 90% for a certain time,
3) If the queue length of any gateway is at Q^{max} for a certain number of consecutive tasks.

The normal operation of *Inter-Gateway Management* is through periodic updates, but irregular interventions may be needed under the given circumstances. If there is persistently not enough bandwidth left or the gateway queue is full at a local network, then the corresponding gateway sends an *emergency signal* to the *Inter-Gateway Management* component. A *reassign & reroute* signal is sent back to gateways that is further forwarded to edge devices. Since gateway assignment is based on fairness, it balances out bandwidth and queue utilizations across local networks.

If there is a failure in the execution of *Inter-Gateway Management*, the Intra-Gateway Management can continue working since the gateways already know their edge device assignments. *Intra-Gateway Management* is a local management scheme, meaning that it does not need to receive external inputs to operate. Each gateway only needs to know their new assignments whenever a there is a restructuring in the network. There are N different *Intra-Gateway Management* instances running at the same time on different gateways separately. On the other hand, if any of the *Intra-Gateway Management* schemes fails, then the *Inter-Gateway Management* can continue operating as well. It can still decide on gateway assignments and routing. However, the failed *Inter-Gateway Management* will not be able to produce optimal offloading rate values, so the performance of the overall management may decrease. The common point in both management mechanisms is that they do not fully rely on a single device to run. There are distributed components that run at edge devices, which significantly reduces the single-point failure phenomena that centralized systems have.

VI. Intra-Gateway Management

We consider two approaches: centralized Model Predictive Control (MPC) and our distributed solution for the control of the local networks. First, we analyze the centralized MPC and discuss its limitations for practical implementations in large networks. It requires to communicate and use the full knowledge of the entire local network, which is not a scalable approach. Thus, we use it as a benchmark to represent the ideal performance. We then 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.

A. Centralized MPC

Our problem in Equation (19) can be converted to the standard MPC form using the following discrete-time prediction model:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{C}\mathbf{w}(k) \tag{23}$$

with state $\mathbf{x}(k) = [q_{A,1}(k), q_{B,1}(k), T_{ed,1}(k), \mu_1(k), L_1(k), ..., q_{A,N_j}(k), q_{B,N_j}(k), T_{ed,N_j}(k), \mu_P(k), L_{N_j}(k), Q_G(k)]^T$and control input $\mathbf{u}(k) = [\Delta f_1(k), \Delta L_1(k), ..., \Delta f_{N_j}(k), \Delta L_{N_j}(k)]^T$. Disturbance vector includes the ambient temperatures of edge devices and gateway's processing rate, which are uncontrollable: $\mathbf{w}(k) = [T_{amb,1}(k), ..., T_{amb,N_j}(k), \mu_G(k)]^T$. For a local network with N_j edge devices, the state, input, and disturbance vectors are of sizes $5N_j{+}1$, $2N_j$, and $N_j{+}1$ respectively. QoS and reliability constraints can be represented as $\mathbf{D}\mathbf{x}(k) \leq 0$ and $\mathbf{E}\mathbf{u}(k) = 0$ in matrix form.

At decision instant k, the controller samples the state of the system $\mathbf{x}(k)$ and solves the centralized optimization problem $Pr_C(\mathbf{x}(k))$ of the following form to find the control action.

$$\begin{aligned} &\min_{\mathbf{u}(k)} \quad \sum_{l=0}^{T_p-1} \sum_{i=1}^{N_j} d(\mathbf{x}(k+l|k), \hat{\mathbf{x}}) \\ &\text{s.t.} \quad i = 1, ..., N_j \quad j = 0, ..., T_p - 1 \\ &\mathbf{x}(k+l+1|k) = \mathbf{A}\mathbf{x}(k+l|k) + \mathbf{B}\mathbf{u}(k+l|k) \\ &\mathbf{D}\mathbf{x}(k+l|k) \leq 0 \\ &\mathbf{E}\mathbf{u}(k+l|k) = 0 \end{aligned} \tag{24}$$

The double index notation $(k{+}l|k)$ in (24) denotes a prediction for l steps ahead from time k. $d(\mathbf{x}, \hat{\mathbf{x}})$ denotes a distance metric. The problem is solved for a prediction horizon of T_p. For centralized MPC, we first set $k = 0$ and find a solution to $Pr_C(\mathbf{x}(k))$, then apply control $\mathbf{u}^*(k|k)$ to the system. Next, k is incremented and the previous steps are repeated until the final horizon T_f.

The centralized MPC approach requires communication of states from all nodes to a central entity (gateway), which then sends an individual control signal to each of the edge devices. The gateways should solve a problem with $(5N_j{+}1){\times}T_p$ states and produce a control sequence of size $2N_j{\times}T_p$ at each time step. Hence, as the network size grows, the computation time required to solve the optimization problem becomes very large. The problem is also a Nonlinear MPC problem because of the nonlinear relationship between the control variables and the objective and constraints, which further exacerbates the computational complexity. The numerical solution of the NMPC optimal control problems is typically based on direct optimal control methods using Newton-type optimization schemes. Even the computational complexity of very low-complexity implementations of NMPC are at least $\mathcal{O}(T(n_x^2{+}n_x n_u))$ [59], with $T = T_p$ being the prediction horizon and $n_x = 5N_j + 1$ and $n_u = 2N_j$ respectively the state and input dimensions for our problem. Finally, the centralized approach is inflexible, in the sense that adding new devices to the network requires the controller to drastically update its model. To address these issues, we distribute the computation among the network devices.

B. Proposed Controller Methodology

We decompose the central Nonlinear MPC problem $Pr_C(\mathbf{x}(k))$ into a set of local subproblems $Pr_p(\mathbf{x}_p(k))$, $p \in \{1, \ldots, P\}$ for the edge devices and a light-weight central subproblem $Pr_G(\mathbf{x}_G(k))$ for the gateway. The goal of this decomposition is twofold: first, to ensure that the central subproblem is computationally much less intensive and smaller in size than the overall problem (has fewer state variables and constraints, and *linear* unlike $Pr_C(\mathbf{x}(k))$), and second, to ensure that the coupling between local subproblems are minimal and solvable in tolerable time in constrained edge devices.

Handling the Size. In our problem, it is redundant to search for an optimal solution over a space of size $(5N+1)\times T_p$ as in Equation (24). The reason is that if the overall system consists of subsystems whose time constants are far from each other (e.g. temperature $T_{ed,i}$ and performance $\{\mu_i, L_i\}$), then the fast varying subsystem (performance) will arrive at its steady-state before the slow subsystem (temperature) has deviated significantly. Leveraging this, we can employ different control periods for the slow and fast subsystems. If the control period of the slow subsystem is longer than the settling time of the fast subsystem, the fast subsystem can always enter its steady-state. Thus, the control loops for them are decoupled and can be designed independently. We decrease the overall problem size by employing larger control periods for slower changing subsystems and separating their control loops.

Handling the Nonlinearity. The "causal chain" of *Frequency → DissipatedPower → Temperature* can be split into two parts. The first part, as expressed by Equation (2), is highly nonlinear while the power-to-temperature model in Equation (6) is linear. We separate the linear and nonlinear parts to keep the MPC model in the central subproblem linear, minimizing its complexity.

Handling the Couplings. Since the states of any edge device pair $\{\mu_i, T_{ed,i}\}$ and $\{\mu_j, T_{ed,j}\}$, $i \neq j$ are already decoupled, a natural way to decompose the problem is to associate local subproblems with only these states to each edge device. The state for communication rates, L_i, are coupled through the gateway queue structure (16) and bandwidth constraints. The battery states $q_{A,i}$ and $q_{B,i}$ are coupled through the objective function (18) that aims at maximizing the battery remaining energy in the edge devices. Therefore, a complete decentralization is not possible and coordination between edge devices is needed. We associate a central subproblem with the coupled states $\{L_i, q_{A,i}, q_{B,i}\}$ to be solved at the gateway using MPC.

C. Proposed Controller Architecture

In the following section, we describe the structure of our proposed controller. Fig. 11 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

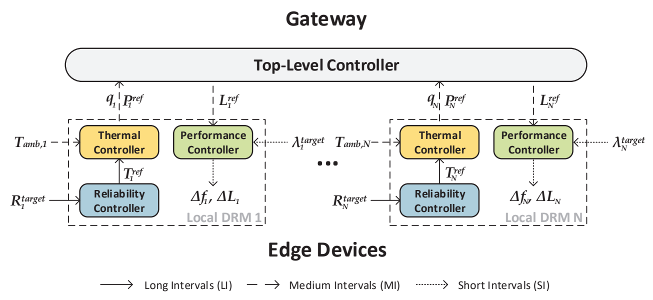


Fig. 11: Controller block diagram

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 [60]. The proposed controller architecture consists of the following four components.

1) Edge 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.

D. Edge Reliability Controller

Leveraging the equivalent degradation time technique in Equation (14), 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 (25). The computation of convex optimization introduces a negligible overhead since the controller activates by intervals in the order of days.

$$\min_{T_{ed}^{target}} \quad \|R(t_{eqv}+t_{rem}, T_{ed}^{ref})\| \tag{25}$$
$$\text{s.t.} \quad R(t_{eqv}+t_{rem}, T_{ed}^{target}) \geq R^{target}$$
$$R(t_{eqv}, T_{ed}^{target}) = R_{d,ed}(k_{LI})$$

$R_{d,ed}(k_{LI})$ 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}^{target}, is the temperature which would satisfy the reliability target R^{target} at the desired lifetime t_{life}, given the device operates at that temperature for the remaining of its lifetime. The constraint on reliability is met if the average LI temperature T_{ed}^{LI} is below T_{target}^{ed} at the end of the LI.

Within an LI, if the difference between target temperature and average temperature, i.e., $T_{ed}^{target} - T_{ed}^{avg}$, is non-zero at any given time instant, then the system has either not fully exploited the available reliability margin (if positive) or it has violated the reliability constraint for the current LI so far (if negative). Therefore, we introduce a new variable T_{ed}^{ref} to keep track of under/over-utilization of the reliability margin and adjust the system accordingly.

$$T_{ed}^{avg}(k_{MI}) = \frac{(k_{MI}-1)\cdot T_{ed}^{avg}(k_{MI}-1) + T_{ed}^{LI}(k_{MI})}{k_{MI}} \tag{26}$$

$$T_{ed}^{ref}(k_{MI}) = \frac{t_{LI}\cdot T_{ed}^{target} - k_{MI}\cdot T_{sys}^{avg}(k_{MI})}{t_{LI} - k_{MI}} \tag{27}$$

In the above equations, k_{MI} indicates the k^{th} MI inside an LI and t_{LI} is the duration of an LI (measured in number of MIs). If the system is being over-utilized, then T_{ed}^{ref} will be lower than T_{ed}^{target}, accommodating for the extra thermal stress experienced until that point. This way, the *Edge Thermal Controller* can reduce the thermal stress for the remaining part of the current LI using T_{ed}^{ref} as a reference.

E. 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}. We recast the temperature state-space model in Equation (6) as follows:

$$T_{ed}(k_{MI}+1) = A_T \cdot T_{ed}(k_{MI}) + B_T \cdot u_T(k_{MI}) \tag{28}$$
$$u_T(k_{MI}) = P_{ed}(k_{MI}) + C_T/B_T \cdot T_{amb}(k_{MI}) \tag{29}$$

Then, the state feedback $u_T(k_{MI})$ is calculated as We apply the state-feedback control law [61] for a linear system. Then, the input $u_T(k_{MI})$ is calculated as:

$$u_T(k_{MI}) = K_T(T_{ed}^{ref} - T_{ed}(k_{MI})) \tag{30}$$

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. Hence, we retrieve $P_{ed}(k_{MI}) = [P_{pu}(k_{MI}), P_{rf}(k_{MI})]^T$ using the following equation.

$$P_{ed}(k_{MI}) = K_T(T_{ed}^{ref} - T_{ed}(k_{MI})) - C_T/B_T \cdot T_{amb}(k_{MI}) \tag{31}$$

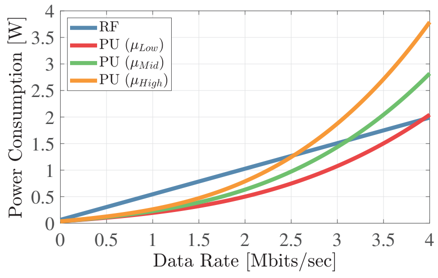


Fig. 12: Average power as functions of computation and communication rates

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 (31) concerning the power scaling of the components P_{pu} and P_{rf}.

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         $P_{ed}^{ref} = [P_{pu}|_{\mu=\mu^{ref}}, P_{rf}(k_{MI})]^T$
```

P_{ed}^{ref} *Trimming.* Up until a certain rate, processing data on the PU consumes less power than communicating the data, as shown in Fig. 12. However, it is more energy efficient to communicate data for higher rates since 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 2 to find a tighter, more energy efficient power reference than what we have obtained in Equation (31) 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 (31). 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).

F. 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. It solves a standard quadratic programming (QP) form MPC with a linear system model.

$$\min_{\mathbf{P}_{ed}} \sum_{k_{MI}=0}^{T_p-1} -\|\mathbf{1}^T\mathbf{q}(k_{MI})\|_Y^2 + \|\mathbf{P}_{ed}(k_{MI}) - \mathbf{P}_{ed}^{ref}\|_Z^2 \quad (32)$$

$$\begin{aligned}
\text{s.t.}\quad & k_{MI} = 0, ..., T_p \\
& q_{A,i}(k_{MI}{+}1) = A_q q_{A,i}(k_{MI}){+}B_q q_{A,i}(k_{MI}){-}\frac{\mathbf{1}^T P_{ed,i}(k_{MI})}{V_{dd}} \\
& q_{B,i}(k_{MI}{+}1) = C_q q_{A,i}(k_{MI}){+}D_q q_{B,i}(k_{MI}) \\
& Q_G(k_{MI}{+}1) = [Q_G(k_{MI}){+}\sum v^T P_{ed,i}(k_{MI}){-}\mu_G(k_{MI})]^+ \\
& Z_{G,j}(k_{MI}{+}1) = [Z_{G,j}(k_{MI}) - \mu_{G,j}(k_{MI}) + \epsilon_{G,j}]^+ \\
& Q_G(k_{MI}) \le Q^{max} \\
& Z_G(k_{MI}) \le Z^{max} \\
& \sum v^T P_{ed,i}(k_{MI}) \le BW
\end{aligned}$$

where we used the fact that $[0\ g/\rho]\cdot P_{ed} = L$ and rewritten the communication rate variables L in terms of power variables P_{ed} with $v^T = [0\ g/\rho]$. $\mathbf{q} = [\mathbf{q}_A, \mathbf{q}_B]^T$ 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 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 $P_{ed}(k_{MI})$ 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*.

G. 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*.

$$\Delta f(k_{SI}){=}K_{P,1}[(\lambda_{target}(k_{MI}){-}L_{ref}(k_{MI})){-}f(k_{SI})] \quad (33)$$

$$\Delta L(k_{SI}){=}K_{P,2}(L_{ref}(k_{MI}){-}L(k_{SI})) \quad (34)$$

VII. Inter-Gateway Management

The optimization problem posed in Section IV-B is Mixed-Integer Programming (MIP). The majority of MIP problems are NP-hard, exact solutions result in poor scalability, and therefore encouraging the use of efficient heuristics to approximate the optimum within finite time. We observe that Equation (22) can be precisely decomposed into its integer and continuous variables. We propose a two-step solution that separates and individually handles these variables:

(i) Determine the sink for each commodity, i.e., find the set of devices $ED_i \in O_j$ for each gateway. The result of this step imposes constraints for the next step; the amount of data absorbed by the gateways for all commodities $r_{kl}^i,\ l \in V_G$.

(ii) Solve the resulting optimization problem over a convex set of continuous variables to find the optimal $\mathbf{r} = \{r_{kl}^i\}$, given the constraints from the previous step.

The first step is essentially a combinatorial problem with the goal of identifying the best edge device to gateway assignment over a finite set of options. The second step consists of only continuous variables, it can be further converted to Linear Programming (LP) for which we explain the procedure below in detail.

A. Gateway Assignment

In Section IV-B we assumed that in general, each commodity should only be communicated to a single gateway, that is, data from one edge device cannot be distributed to multiple gateways. A gateway assignment step is necessary as a result of this assumption. However, in some circumstances, it may be admissible to forward any commodity to any gateway. For example, if the gateways are interconnected via Ethernet, they can reshare the offloaded data over Gbps-speed wired connections with minimal delay. Or, if any of the gateways can carry out the same type of tasks with the received data, there is no need to try forwarding data exclusively to a particular one. For such cases, our approach offers natural way of separating the overall problem; if the network allows for communication to any gateway,it is sufficient to solve only step (ii), bypassing the gateway assignment step.

We assign edge devices to gateways primarily based on fairness: each gateway should receive similar amounts of offloaded data and in proportion to their available computational resources. The bandwidth is limited at each local network (per gateway). Furthermore, we try to allocate less data to the gateways with higher queue utilization and assign edge devices to closer gateways in terms of physical distance. The gateway assignment problem is formulated as follows:

$$\begin{aligned} \underset{\mathbf{X}_{assign}}{\text{maximize}} \quad & \min_{j \in V_G} \sum_{i=1}^{N} c_{ij} x_{ij} \qquad (35) \\ \text{subject to} \quad & \sum_{i=1}^{N} L_i x_{ij} \leq BW_j, \ \forall j \in \{1, ..., M\} \\ & \sum_{j=1}^{M} x_{ij} = 1, \ \forall i \in \{1, ..., N\} \\ & x_{ij} \in \{0,1\}, \ \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., M\} \end{aligned}$$

where $\mathbf{X}_{assign}$ is the assignment matrix in which elements x_{ij} assume value 1 if edge device i is assigned to gateway j and 0 otherwise. This is a problem from the class of bottleneck generalized assignment problems (BGAP) [62] and many heuristic and exact solution procedures exist [63], [64], [65]. The GAP is a well-known integer programming problem involving the assignment of a number of jobs to a number of agents such that each job is performed by a unique agent, capacity limitations on the agents are not exceeded, and the total cost of the assignments is minimized. The bottleneck (or minimax) version of this problem is where the objective is to minimize the maximum of the costs of the assignments that are made. In our problem, there are *N* commodities (tasks) that are assigned to *M* gateways (agents). The offloading rates L_i of the corresponding commodities are the number of resource units consumed. c_{ij} is the cost of gateway j to consume commodity of edge device i, which we define as a decreasing function of queue utilizations Q_i and distance d_{ij}.

We use the approximate algorithm in [64] that heuristically searches for a solution to BGAP. The algorithm is centralized. All edge devices communicate the values of their offloading rates to a head gateway. Then, the problem is solved to find the optimal assignments and the results are communicated back to the edge devices.

B. Routing

The gateway assignment step specifies the amount of data to be absorbed by the gateways for all commodities. The remainder of the overall problem is then a multicommodity flow multiple sink routing problem with known commodity-to-sink assignments. From this point, we solely need to deal with continuous functions defined on a set of continuous variables, that is, the flow rates.

The MTTF function itself is non-linear and non-convex, still, the optimization problem can be linearized in a few steps. We first start by taking the natural logarithm of the objective function.

$$\log MTTF_{ed,i}(\mathbf{r}) = \log \gamma_c + \frac{E_a}{k_B T_{ed,i}(\mathbf{r})} \qquad (36)$$

Maximizing the minimum of $\log MTTF_{ed,i}$ is an equivalent problem to our original problem. The decision variable, flow rate vector $\mathbf{r}$, is related to MTTF through temperature function $T(\mathbf{r})$. From Equation (6), temperature is a function of power dissipation. On the other hand, power is a function of data flow rates through Equation (3). Both relations are linear, but the temperature equation is time-dependent. To have a time-invariant approximation for device temperature, we use the state-space formulation and find the time step t_{ss} where temperature reaches a steady-state. We unroll the list of linear equations until time step $t = t_{ss}$ and calculate two coefficients k_1 and k_2 of power P and ambient temperature T_{amb} respectively. The time-invariant temperature equation used in our problem formulation is as follows:

$$T_{ed,i}(\mathbf{r}) = k_1 P_{ed,i}(\mathbf{r}) + k_2 T_{amb,i} + k_3 \qquad (37)$$

This can be explicitly written as $T_{ed}(\mathbf{r}) = \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i$ in summation form where E is a matrix. The entries E_{kl} are constants depending on the pair of nodes k and l, while the bias terms in (37) are omitted without loss of generality. We also do not show the constant term $\log \gamma_c$ in the following derivation to further simplify notation. Then, $\log MTTF_{ed,i} \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i = \frac{E_a}{k_B}$.

Altogether, the problem in Equation (22)–excluding the gateway assignment component– can be rewritten as

$$\begin{aligned} \text{maximize} \quad & MTTF \qquad (38) \\ \text{subject to} \quad & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = L_i, \ i = k \ \forall i, k \in V_{ED} \\ & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = 0, \ i \neq k \ \forall i, k \in V_{ED} \\ & r_{kl}^i \geq 0, \ \forall i, k \in V_{ED}, \ \forall l \in S_k \\ & \sum_{k \in V_{ED}} r_{kl}^i = L_i, \ \forall i \in V_{ED}, \ \forall \{l \mid ED_i \in O_l\} \\ & MTTF \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i \leq \frac{E_a}{k_B}, \ \forall i, k \in V_{ED} \end{aligned}$$

The last set of inequality constraints combined with the new objective variable ensures that the minimum MTTF of all nodes in the network is maximized. We convert this problem into an equivalent linear programming formulation by change of variables $y = 1/MTTF$.

$$\begin{aligned} \text{minimize} \quad & y \qquad (39) \\ \text{subject to} \quad & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = L_i, \ i = k \ \forall i, k \in V_{ED} \\ & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = 0, \ i \neq k \ \forall i, k \in V_{ED} \\ & r_{kl}^i \geq 0, \ \forall i, k \in V_{ED}, \ \forall l \in S_k \\ & \sum_{k \in V_{ED}} r_{kl}^i = L_i, \ \forall i \in V_{ED}, \ \forall \{l \mid ED_i \in O_l\} \\ & \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i \leq y \frac{E_a}{k_B}, \ \forall i, k \in V_{ED} \end{aligned}$$

We can interpret the above problem as minimizing the upper bound q on the inverse of the mean time to failure of all nodes in the network. Following a similar rationale as discussed in Section VI-A, we propose to solve this problem in a distributed manner. A centralized solution is not desirable due to lacking scalability and flexibility. Therefore, we decompose the problem into subproblems with dual decomposition, then solve them distributedly at each node using the subgradient method.

Distributed Algorithm. We first convert the problem in Equation (38) into a completely decomposable form by introducing additional variables. The objective function y is replaced by $\sum_{i \in V_{ED}} y_i^2$, similar to the technique presented in [66]. Under this new objective function, network lifetime optimization is reformulated as a quadratic programming problem.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in V_{ED}} y_i^2 \qquad (40)\\
\text{subject to} \quad & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = L_i, \; i = k \; \forall i, k \in V_{ED}\\
& \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = 0, \; i \neq k \; \forall i, k \in V_{ED}\\
& r_{kl}^i \geq 0, \; \forall i, k \in V_{ED}, \; \forall l \in S_k\\
& \sum_{k \in V_{ED}} r_{kl}^i = L_i, \; \forall i \in V_{ED}, \; \forall \{l \mid ED_i \in O_l\}\\
& \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i \leq y_k \frac{E_a}{k_B}, \; \forall i, k \in V_{ED}\\
& y_i = y_j, \; \forall i \in V_{ED}, \forall j \in S_i
\end{aligned}$$

Here, we have local variables y_i's for each node and constraints that enforce them to be equal. The objective function is quadratic and thus strictly convex in the y_i's. We need to find a flow $\mathbf{r}$ minimizing this objective, such that $\frac{1}{y_i} \leq MTTF_{ed,i}(\mathbf{r})$, which can be done using the dual decomposition approach.

We construct the dual problem by introducing Lagrange multipliers υ_i for the flow conservation constraints and ν_{ij} for the lifetime equality constraints. This results in *Partial Lagrangian* given by (41), where the linear equality constraints (temperature constraints) are not relaxed as they can be satisfied locally at each node. We also do not include the single gateway communication constraint $\sum_{k \in V_{ED}} r_{kl}^i = L_i$. Instead, we manually set the flows $r_{kl}^i = 0$ for all $i, k \in V_{ED}$ and all $l \in V_G$ such that $ED_i \notin O_l$. This ensures there are no flows to other gateways and all the flow is restricted to be communicated to the assigned one.

$$\begin{aligned}
L(\mathbf{y}, \mathbf{r}, \upsilon, \nu) = & \sum_{i \in V_{ED}} y_i^2\\
& + \sum_{k \in V_{ED}} \sum_{\substack{i=1\\ i=k}}^{N} \upsilon_k^i \Big\{ \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) - L_i \Big\}\\
& + \sum_{k \in V_{ED}} \sum_{\substack{i=1\\ i\neq k}}^{N} \upsilon_k^i \Big\{ \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) \Big\}\\
& + \sum_{i \in V_{ED}} \sum_{j \in S_i} \nu_{ij} (y_i - y_j)\\
= & - \sum_{k \in V_{ED}} \sum_{\substack{i=1\\ i=k}}^{N} \upsilon_k^i L_i + \sum_{k \in V_{ED}} \Big\{ y_k^2\\
& + y_k \sum_{l \in S_k} (\nu_{kl} - \nu_{lk}) + \sum_{i=1}^{N} \sum_{l \in S_k} r_{kl}^i (\upsilon_k^i - \upsilon_l^i) \Big\}
\end{aligned} \qquad (41)$$

The dual function is given by

$$g(\upsilon, \nu) = \inf_{\mathbf{r}, \mathbf{y}} \left\{ L(\mathbf{y}, \mathbf{r}, \upsilon, \nu) \;\middle|\; \begin{array}{l} 0 \leq r_{kl}^i, \; \forall i \in V_{ED}, \forall l \in S_k \\ \sum\limits_{l \in S_k} E_{kl} \sum\limits_{i=1}^{N} r_{kl}^i \leq y_k \frac{E_a}{k_B}, \; \forall i, k \in V_{ED} \end{array} \right\} \qquad (42)$$

From the expression of the Lagrangian, it is clear that the dual function can be evaluated separately in the variables corresponding to each node $k \in V_{ED}$. The variables local to node k are y_k and r_{kl}^i, $l \in S_k$. We use the subgradient method [67], [68] to solve dual problem in a distributed manner.

Subgradient Method: It is an iterative optimization algorithm for minimizing nondifferentiable convex functions. At each iteration t, the nodes only use the local information available E_{kl} and the Lagrange multipliers $\upsilon_k(t)$, $\nu_{kl}(t)$ to solve the following convex quadratic program with variables $y_k(t)$, $r_{kl}^i(t)$ for $k \in V_{ED}$, $l \in S_k$.

$$\begin{aligned}
\text{minimize} \quad & y_k^2(t) + y_k(t) \sum_{l \in S_k} (\nu_{kl}(t) - \nu_{lk}(t)) \qquad (43)\\
& + \sum_{i=1}^{N} \sum_{l \in S_k} r_{kl}^i(t) (\upsilon_k^i(t) - \upsilon_l^i(t))\\
\text{subject to} \quad & \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i(t) \leq y_k(t) \frac{E_a}{k_B}\\
& r_{kl}^i(t) \geq 0, \; \forall i \in V_{ED}, \; \forall l \in S_k
\end{aligned}$$

The optimal values of the above problem are then used to evaluate the subgradient components of $-g$ for given (υ,ν) pair at iteration t. The subgradients are given by

$$\mathrm{f}_k^i(t) = \begin{cases} L_i - \sum_{l \in S_k} (r_{kl}^i(t) - r_{lk}^i(t)), \; i = k \\ - \sum_{l \in S_k} (r_{kl}^i(t) - r_{lk}^i(t)), \; i \neq k \end{cases}$$

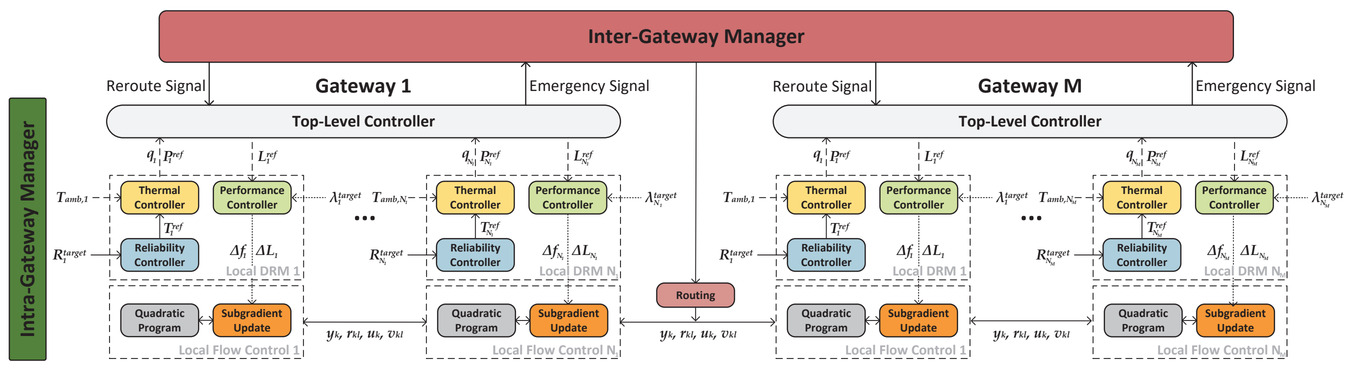


Fig. 13: Framework block diagram

$$h_{kl}(t) = y_l(t) - y_k(t)$$

Finally, the Lagrange multipliers $\upsilon_k(t)$ and $\nu_{kl}(t)$ are updated using the subgradients based on the following equations:

$$\upsilon_k^i(t+1) = \upsilon_k^i(t) - \beta(t) f_k^i(t) \quad (44)$$

$$\nu_{kl}(t+1) = \nu_k(t) - \beta(t) h_{kl}(t) \quad (45)$$

where $\beta(t)$ is a positive scalar step-size. After solving problem (43) and updating the Lagrange multipliers, each node exchanges the updated values of y_k, r_{kl}, υ_k, and ν_{kl} with their neighbors $l \in S_k$.

VIII. Evaluation

A. Experimental Setup

To illustrate the effectiveness of our solution, we conduct experiments on realistic edge computing scenarios. In our simulations, we use real power and temperature measurements collected from actual IoT devices. The experiments are realized on MATLAB. Fig. 13 shows the block diagram of the simulation infrastructure. Some important model parameters used in the simulation are summarized in Table

TABLE II: Model Parameters

Parameters	Value	Parameters	Value
a	1.59×10^{-11}	g	0.12×10^{7}
b	8.62×10^{-7}	SI	0.1 s
C_f	0.22	MI	1 s
C_s	1×10^{-4}	LI	86,400 s
β	3.2	T^{ref}	$45°C$
ρ_2	0.064 W	ϵ_G	10

Hardware: The target edge devices are Raspberry Pi 2 with ARM Cortex-A7 CPU and the gateway is a Raspberry Pi 4 Model B with Arm Cortex-A72 CPU. The gateway (Pi 4) exhibits around ten times more instructions per second compared to edge devices (Pi 2) [69]. We measure the CPU and WiFi power consumption and temperature of the edge devices by running various applications under different ambient temperatures, then fit the models in Section III. The maximum power consumptions of PU and RF components are $P_{pu}^{max} = 2.16W$ and $P_{rf}^{max} = 1.44W$ respectvely. We use the *T-KiBaM* [45] 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 (12) 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 [70], 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 [14] with their corresponding power consumptions. These classification and regression tasks can either run on the edge devices or the gateway. In the simulations we assign tasks in a randomized fashion, to immediately run one after another. The task sizes are sampled from an exponential distribution, where the mean size is 5 MB. Task are completed when all the data belong to a task is processed either by edge device or gateway. We randomly pick delay-sensitive or delay-tolerant tasks from the whole set of tasks. For delay-sensitive tasks, the task deadlines D_m are assigned randomly from a uniform distribution, $U(0.2, 2)$ seconds. The offloaded data for a given task should not wait in the gateway queues longer than the assigned deadline.

We conduct experiments based on a practical scenario of human activity recognition (HAR) [71], implemented on edge devices [50]. 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 [50]. We randomly assign the data input rate λ^{target} for a given task and choose it from a uniform distribution, $U(0,1)$ Mbps. Furthermore, we assume that each gateway can allocate $\mu_G = 3$ Mbps processing rate for the offloaded data, deterministically, constant throughout the experiments.

Topology: In the following, we first present results on an example of local network with a single gateway and solely demonstrate the performance of the *Intra-Gateway Management* piece of our approach. We then consider a large-scale example with multiple gateways for thorough evaluation. For both scenarios, we set the bandwidth limit to be $BW = 5$ Mbps for each local network (per gateway).

B. Local Network - Single Gateway Results

We perform simulations on a network with single gateway and 8 edge devices randomly distributed over a field of 50m x 50m, with $d_{max} = 25$. We compare our *Intra-Gateway Management* approach with the following techniques:

- *No ERC-ETC* is our solution without the *Edge Reliability Controller (ERC)* and *Edge Thermal Controller (ETC)*.
- *All Edge* is the naive approach which 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 [11]. 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 offloading, the edge devices are allocated more communication bandwidth.
- *Pagliari* (the name of the author) [21] 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 gains of adopting our solution. The target reliability for the edge devices is empirically selected to be 0.85 at t_{life} of 3 years (36 months). Values ranging from 0.6 to 0.9 are commonly selected as the cut-off levels for 3 to 5 years of target lifetime [39]. Fig. 14 shows the time it takes for the edge device with the *minimum* reliability in the network to violate 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 fail much sooner, falling short by as much as 20 months, 7 months being the best. Fig. 15 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.

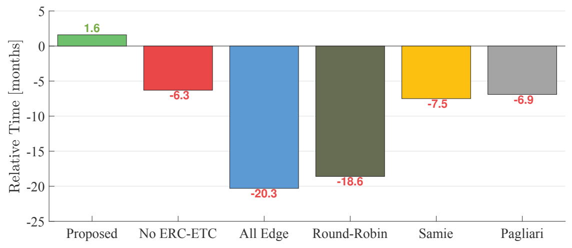


Fig. 14: The relative time until edge device violates the reliability target of 0.85 (positive is better)

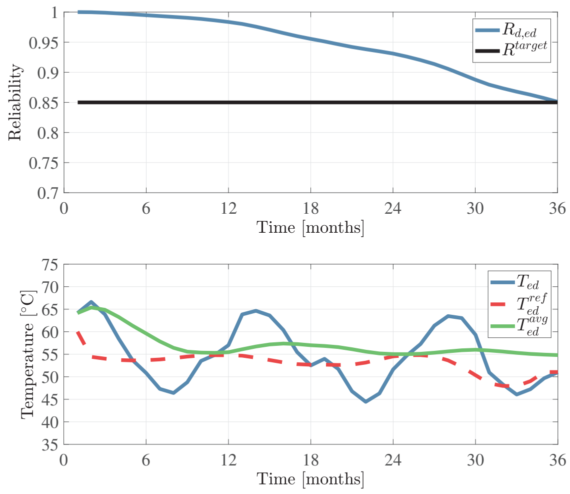


Fig. 15: Reliability curve and the long time temperature behaviour of the device and the controller

Energy Savings: Fig. 16 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 requirements. *Samie* [11] 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 as can be seen in Fig. 14.

Quality of Service: As described in Section IV, there are three QoS constraints: input data rate, network bandwidth, and

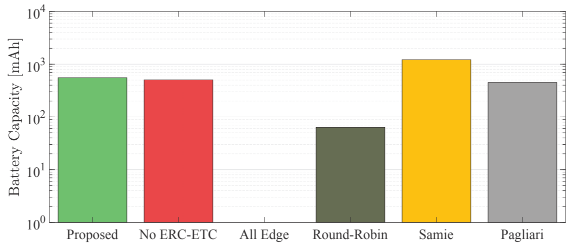


Fig. 16: Minimum battery charge in the network

task deadlines. The proposed controller satisfies all of them for a single local network. There are no violations due to the strict constraints in the MPC controller employed. A detailed evaluation is carried out for multi-gateway networks in the following subsection.

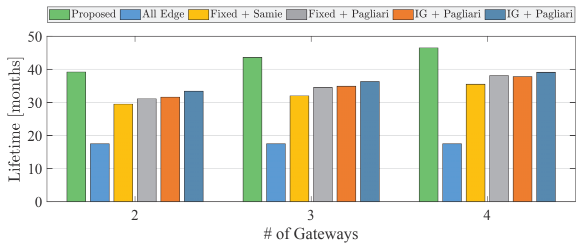
C. Multi-Gateway Network Results

We further evaluate our proposed solution including both the *Intra-Gateway Management* and the *Inter-Gateway Management* components on multi-gateway networks. The experiments are repeated for different number of gateways: 2, 3, 4, and for different number of edge devices: 12, 36. Network devices are assumed to be randomly distributed over a field of 100m x 100m, with $d_{max} = 50$. All other parameters and variables are kept the same as for the local network simulations.

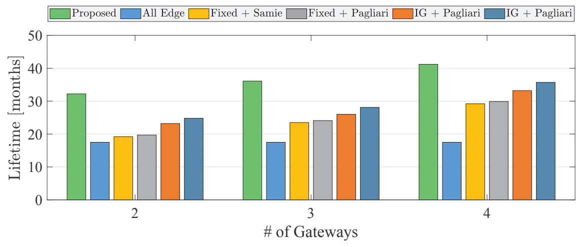
None of the comparisons in the previous section were proposed for multi-gateway systems, so we modified them by adding routing and gateway selection capabilities. We select three baseline methods: *All Edge*, *Fixed + Samie*, and *Fixed + Pagliari*. The prefix label *Fixed* means that the routes are static and the topology is fixed. We assign N/M edge devices to their closest gateways and the assignments do not change over the simulation horizon. We pick *Samie* [11] and *Pagliari* [21] for further evaluation as the former was the approach providing the best battery lifetime and the latter was the best reliability comparison. For more elaborate testing, we also implement them on top of our *Intra-Gateway Management* solution for routing and gateway assignment, denoted by *IG*. This helps us single out and show the contribution of *Intra-Gateway Management* when compared with the *Fixed* versions of the same methods. The evaluated approaches are summarized below.

- *All Edge* assigns all the computation to the edge devices with no offloading to the gateways, only the output of the compute processes are communicated to the gateways.
- *Fixed + Samie* is Samie's approach with fixed topology.
- *Fixed + Pagliari* is Pagliari's with fixed topology.
- *IG + Samie* is Samie's approach with our routing and gateway selection method added.
- *IG + Pagliari* is Pagliari's approach with our routing and gateway selection method added.

Reliability: Similar to the local network simulations, the target reliability for edge devices is selected to be 0.85 at 36 months. In other words, the degradation in the reliability of edge devices should not exceed 0.15 to achieve desirable MTTF. Fig. 17a illustrates how long it takes for the edge devices to degrade below this desired value. The results are given for a network of 12 edge devices, but only the minimum lifetime amongst those is plotted. The target reliability is reached in 39.2, 43.2, and 46.5 months with our approach for 2, 3, and 4 gateways respectively. All approaches follow the same trend with an improvement in lifetime for increasing number of gateways. When there are more gateways, offloaded data needs to travel less, either in terms of the number of routing hops or the actual physical distance. Moreover, edge devices can offload more data because bandwidth occupation



(a) 12 edge devices



(b) 36 edge devices

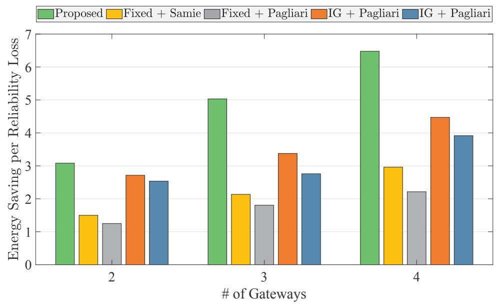
Fig. 17: The lifetime until reliability target violation

and queue lengths at the gateways are reduced. The proposed approach delays reliability violation by 5.8 months compared to the closest approach (*IG+Pagliari*) for 2 gateways to as much as 7.4 months for 4 gateways. Except the proposed approach, all approaches fail to meet the target time of 36 months for the configuration with 2 gateways. We also observe that introducing *Inter-Gateway Management* to other approaches improves lifetime. For example, *Pagliari* approach gains 2.3 months with *IG* over fixed gateway assignment and routing.

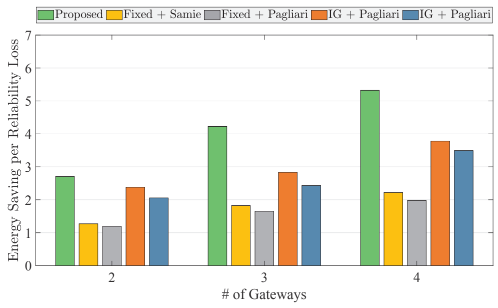
For the network with 36 edge devices, as shown in Fig. 17b, edge devices degrade at higher rates and reliability target is violated sooner. The bandwidth gets occupied much faster, queues are filled up, and edge devices can offload much less data to the gateways. As a result, all approaches tend to behave more like the *All Edge* approach, because most of the processing should be done at the edge devices in the lack of offloading opportunities. In this case, the performance gap between the proposed approach and others is widened, displaying at least 7.4 months difference for 2 gateways.

Energy Savings: The primary goal in this work is to reduce maintenance costs of IoT systems by means of reliability management. Maintenance costs arise as a result hardware faults, which require repair, component replacement, or complete node replacement. The hardware faults can be attributed to power outages caused by battery depletion and failures due to reliability degradation. Therefore, as in [72], maintenance cost for a network can be formulated as a function of both energy depletion and reliability degradation. It should be noted that one may maximize the time it takes for the batteries to deplete by simply choosing to offload if the communication power consumption is lower than computation for a given task and input data rate. This does not necessarily improve device lifetime or maintenance cost as these decisions can induce higher reliability degradation on the device. Also if there are energy harvesting sources available, then metrics such as battery lifetime do not carry significance as much.

We are interested in an evaluation criteria that covers both



(a) 12 edge devices



(b) 36 edge devices

Fig. 18: Energy savings over reliability degradation

energy savings and reliability degradation. In particular, we want to answer the question: "The energy savings come at the cost of how much degradation in reliability?". Intuitively, the amount of loss in reliability should decrease with increasing energy savings. However, it is not only the amount of energy saved that influences reliability; the timing of savings are critical as well. If the device has high power dissipation during times when its temperature is high (due to ambient conditions), then the effect of this on its reliability will be detrimental. Ideally, energy savings should come when the ambient conditions are severe, and power dissipation should occur when device is cooler. This arrangement would lead up to the least amount of loss in reliability.

In Fig. 18a, we evaluate and show energy savings over degradation in reliability for each approach with a network of 12 devices. The results improve with energy savings and inversely proportional to degradation. We find energy savings relative to the baseline approach *All Edge* where edge devices do all the computation. For example, energy saving value 1.2 means the battery lifetime is 20% improved over running all the workloads on the edge, without any offloading. Reliability degradation is calculated in the standard way used throughout the paper. It can be seen from the plots that the proposed approach provides high energy savings while preserving reliability, up to 49.0% improvement over the closest approach for 3 gateways. This can be attributed to the fact that our approach offloads computation to both reduce thermal stress and save energy, i.e., the savings come at the right time and in the right amount. The difference between *All Edge* and the others is evident with higher number of gateways since offloading becomes much more efficient compared to local processing.

The same procedure is repeated for a network with 36 edge devices and the results are depicted in Fig. 18b. Similar to our reliability results, the improvements decrease when there are more edge devices in the network. For example, the proposed approach is $6.48x$ better than *All Edge* for the network with 12 edge devices and 2 gateways whereas the gain is $5.32x$ for 36 edge devices. Deploying more gateways per edge device can extend the lifetime of the edge devices and reduce their energy consumption.

TABLE III: Quality of Service

	Deadline Misses [%]	Gateway Utilization [%]			
		G_1	G_2	G_3	Avg
Proposed	0.0	90.6	94.1	92.2	92.3
Fixed + Samie	22.1	62.0	99.9	91.3	84.4
Fixed + Pagliari	7.7	54.5	67.5	66.7	62.9
IG + Samie	8.2	96.4	99.1	98.5	98.0
IG + Pagliari	3.8	80.6	87.1	84.0	83.9

Quality of Service: All approaches satisfy the bandwidth constraints since each are strictly forced not to exceed them. We report task deadline miss ratios and gateway utilization values in Table III. The experiments are simulated for a network of 36 edge devices and 3 gateways. Averaged deadline misses and gateway utilization are given over all devices. We define gateway utilization as the percentage of time the gateway is busy. A gateway is assumed busy unless there is no data waiting in its input queue to be serviced. Deadlines are missed if data of a certain task waits in the gateway queue longer than the task deadline.

Our proposed approach does not miss any deadlines because the queueing dynamics and maximum deadline constraints were explicitly considered in the solution. As a naive approach, the tasks can always be executed completely on edge devices at the desired input rates. This would also yield zero deadline misses, but perform the worst in terms of reliability and energy as shown above. The proposed approach offloads data to preserve energy and reliability whenever it is possible to do so without violating deadlines. In such cases when gateway queue lengths grow and execution times approach maximum deadline values, operation is switched to complete local processing to avoid any misses. Other approaches are not deadline-aware and produce misses from 3.8% (*IG + Pagliari*) up to 22.1% (*Fixed + Samie*). Fixed routing and gateway assignments particularly increase deadline misses since it is not possible to reassign edge devices to different gateways when queues are filled up. *Pagliari* [21] approach considers task execution times which improves its performance in comparison to *Samie* [11].

We explicitly report individual gateway utilization values along with their average to show the variation between different gateways in the network. For all approaches the gateways are highly utilized since there is a large number of edge devices per single gateway. It is favorable to utilize the gateways whenever appropriate by offloading data since it reduces the excessive load on the edge devices. Our approach has the second highest average utilization with 92.3% after *IG + Samie* with 98.0%. *Samie* approach iteratively increases offloading and uses more gateway resources if communication is more energy efficient than computation for a given task and data rate combination. This usually results in completely utilizing the gateways until no available resources left, which also leads to deadline misses. In comparison, the proposed approach is more conservative with the gateway use and avoids any adverse overutilization outcome.

The balance of utilization across gateways is an important criteria for QoS as well. For *Fixed + Samie*, some gateways are overloaded (99.9% and 91.3% utilization) with offloaded data despite other gateways being underutilized (62.0% utilization). Such an unbalanced employment of gateway resources would lead to suboptimal QoS; the load on the gateways should be distributed evenly. Our approach and the other approaches assisted by *Inter-Gateway Management* exhibit low utilization variation between different gateways. The maximum deviation is $\pm 3.5\%$, $\pm 2.7\%$, and $\pm 6.5\%$ for *Proposed*, *IG + Samie*, and *IG + Pagliari* respectively. Gateway assignments are based on fairness under *Inter-Gateway Management*, hence, it balances out bandwidth and gateway utilizations across local networks.

Delay-Sensitive Tasks. For the above experiments, we picked delay-sensitive or delay-tolerant tasks randomly from the whole set of tasks. The existence of delay-tolerant tasks helps edge devices to more flexibly offload data as long queue wait times are not a problem for them. Intuitively, it is favorable to serve delay-sensitive tasks at the edge devices whereas offload the delay-tolerant tasks. We conduct further experiments, separately for delay-tolerant and delay-sensitive tasks, to elaborate on this intuition. The experiments are simulated for a network of 36 edge devices and 3 gateways similar to the previous Quality of Service experiments, but we only report results for our proposed approach. For delay-sensitive tasks, we assign the task deadlines D_m randomly from a uniform distribution, $U(0.2, 2)$ seconds.

TABLE IV: Quality of Service for Different Task Types

	Deadline Misses [%]	Gateway Utilization [%]			
		G_1	G_2	G_3	Avg
Delay-Tolerant	0.0	98.6	97.4	97.4	97.8
Delay-Sensitive	0.0	82.6	86.9	85.4	85.0

Table IV presents the deadline miss and gateway utilization results for the two task types. For both, we have 0.0% deadline misses because the tasks can always be executed completely on edge devices at the desired input rates as a naive approach. On the other hand, gateways are utilized more for the delay-tolerant tasks compared to delay-sensitive tasks, with averages 97.8% and 85.0%, respectively. If the queue is already filled and the wait times are higher than task deadlines, then the data of delay-sensitive tasks are processed at the edge device instead of being offloaded.

Packet Loss. Throughout our experiments, the assumption was that the communication is perfectly reliable. Therefore, every packet transmission is assumed successful, i.e., no packet drops. This might not be true in realistic communication scenarios. Here, we consider a more practical scenario, with probabilistic packet losses where retransmissions are handled with a mechanism like TCP. For simplicity, we set a link erasure probability, that is, the probability of losing the complete data that belong to a task, instead of specifying a bit error rate or a packet error rate. Then, data offloading for a task fails with probability p_e. When failure occurs, the complete task data needs to be communicated again. For example, if the task size is 5 Mb and data rate is 1 Mbps, then the offloading of this task is delayed by $\frac{5Mb}{1Mbps} = 5sec$ in case of a failure. We run simulations for various failure probabilities and report corresponding deadline misses in Table V. Only delay-sensitive tasks are used for these experiments with the aforementioned specifications.

TABLE V: Quality of Service Under Packet Loss

p_e	0.0	0.1	0.2	0.3	0.4	0.5
Deadline Misses [%]	0.0	3.3	6.8	17.2	34.5	72.1

As seen from the table, deadline misses are not 0.0% as we impose transmission failures. This is expected as our algorithm does not account for the possible connection errors. When the transmission fails for a task, the queue fills up due to other offloading edge devices until the retransmission starts. Since the proposed method does not include the retransmission delay in the overall delay calculation, we start observing deadline misses.

Low-Capability Edge Devices. According to our edge device model and task input rate specifications, the tasks can always be executed completely on edge devices at the desired input rates. In other words, the edge devices can support up to 1 Mbps data processing rate, which is the maximum λ^{target} value we set for our tasks. As a result of this assumption, deadline misses can be avoided with the naive approach of processing everything on the edge devices with no offloading. Though, it should be noted that this approach severely degrades edge device reliability and consumes excessive energy. We now assume low-capability edge devices that have lower processing data rates than the maximum task input rate. Let us randomly assign the data input rate λ^{target} for a given task and choose it from a uniform distribution, $U(0, 1)$ Mbps as before, but limit the edge device processing rate μ. Below table shows deadline misses for various limits μ_{lim} on edge device processing rates. Only delay-sensitive tasks are used for these experiments with the aforementioned specifications.

TABLE VI: Quality of Service Under Packet Loss

μ_{lim} [Mbps]	-	0.9	0.8	0.7	0.6	0.5
Deadline Misses [%]	0.0	0.0	2.6	4.1	5.8	10.8

The proposed approach can still meet the deadlines perfectly when the edge devices are only capable of processing data with rate 0.9 Mbps. However, deadline misses increase with decreasing μ_{lim} as edge devices necessarily need to offload more data.

Communication & Computation Overhead: We evaluate the overhead of the communication for *Intra-Gateway Management* and *Inter-Gateway Management* on the actual data communication between edge devices and gateways. Let *Mngmt_Data* be the total amount of data exchanged for management and *Task_Data* be the total amount of task-related data offloaded by the edge devices to gateways. Then, the overhead in the occupied communication bandwidth is defined as:

$$\frac{Mngmt\_Data}{Task\_Data} \times 100\% \tag{46}$$

We simulate and log all data exchanges within network devices, then sum the values to find the total amounts for both management-related and task-related data. The experiment is done for the same network as previous subsection, consisting of 36 edge devices and 3 gateways. Data exchanges for management is found to be introducing 2.8% overhead in communication bandwidth. We also measure the overhead in the number of messages communicated, computed as follows:

$$\frac{Mngmt\_Msgs}{Task\_Msgs} \times 100\% \tag{47}$$

where *Mngmt_Msgs* is the total number of messages exchanged for management and *Task_Msgs* is the total number of task-related messages communicated. Here we assume that data for each task is sent in a single message, but in practice it should be packetized. Thus, we essentially measure the number of times a new connection (e.g., a TCP flow) is established between two devices. Results show that the overhead in the number of exchanged messages is 11.2%. Management messages are small because they contain a few values whereas task-related messages is large in volume. Therefore, if the task-related messages are packetized, then they overwhelm the management messages in count.

The *Intra-Gateway Management* requires the communication of power consumption and battery energy values from edge devices to gateways. Then, the gateways communicate back offloading rate values to the edge devices. This needs to be repeated every medium interval (MI). A total of $5N$ real numbers are communicated per MI seconds. For *Inter-Gateway Management*, the subgradient method is fully-distributed and iteratively converges to the solution. Each edge device exchanges the updated values of optimization variable, decision variable, and Lagrange multipliers of the subproblems they are solving with their neighbors. A node is then needs to communicate $3N+M+1$ real numbers per neighbor. In the initial run of the distributed algorithm, it converges to the optimal point after 5000 iterations for a network with 36 edge devices. However, an optimal solution is not necessarily needed as *Intra-Gateway Management* carries out further optimizations. The subgradient method can be terminated within 5% of the optimal value around 1000 iterations. Moreover, the algorithm converges in much fewer iterations, lower than 100, when initialized from the previous solution.

We also discuss the computation overhead of our solution. For *Intra-Gateway Management*, each edge device runs reliability, thermal, and performance controllers. The *Edge Reliability Controller* solves a convex optimization problem. The computation of convex optimization introduces a negligible overhead since the controller activates by intervals in the order of days. The *Edge Thermal Controller* and *Edge Performance Controller* are simply linear state-feedback controllers that can be implemented with a single floating-point dot-product per iteration. The *Gateway Top-Level Controller* solves a standard quadratic programming (QP) form MPC with a linear system model at intervals in the order of a few seconds. For similar scale QP problems to ours, commercial solvers can compute the solution under a millisecond [73]. Finally, we have two separate algorithms under *Inter-Gateway Management*: routing and gateway assignment. Since the routing algorithm is a distributed implementation of a linear programming problem, its computation involves solving only a very small scale convex optimization at each edge device. This incurs a latency in the order of only microseconds. We use a centralized approximate algorithm that heuristically searches for a solution to gateway assignment. This can be computed under a second in modern processors in gateway devices [64]. It should be noted that the frequency of Inter-Gateway Management updates are much lower in comparison.

IX. Discussion

The inherent assumption in our method is that network Mean Time to Failure (MTTF) is the minimum of any node in the network, meaning that the network lifetime is the time until the first node dies. This definition is one of the most prevelant in literature and was used in many recent works [12], [74], [75]. Therefore, we consider the device-level reliability optimization or requirements, and try to keep the reliability of the most degraded device at high levels. Intuitively, this would be ideal for networks where all nodes are equally critical for the system to operate. However, there are interactions between devices and there might be dependencies in data, or between different devices. Furthermore, the devices are heterogeneous. There might be redundancy of devices such as backups, hence a single edge device failure may not result in the failure of operation of the entire network. Other edge devices with sensors of the same or even different types can substitute their work, such that the fault goes undetectable. When all aspects are considered, we need more sophisticated system-level reliability models.

A possible future direction is to incorporate different network reliability models in our management approach. Different from single device, network-level reliability modeling can be examined by graph-based models. For example, a simple model is formulized by the serial reliability expression as described in our paper. Commonly used analytical models for networks include Fault Tree [76], Binary Decision Diagram (BDD) [77], Reliability Block Diagram (RBD) [78], graph transformation [79], or state-space methods such as Markov Chains [80] and Petri Nets [81] — see [82] for a comprehensive survey. The first step in assessing the impact of individual device reliability and failure mechanisms is to determine the conditions for network failure. Furthermore, it is crucial to analyze how these conditions change depending on the IoT application. One can then identify the reliability bottlenecks in the application and reconfigure the management algorithm to adapt. There are strategies that rank the devices' importance within an application and their criticality towards ensuing the network failure condition, based on certain metrics such as Birnbaum's measure [83]. Using techniques such as in [76], [84], we can evaluate the system reliability, determine the criticality of IoT devices, and construct system-level models that reflect inter-dependencies between devices.

X. Conclusion

In this paper, we introduced a dynamic management scheme for IoT edge computing systems. The goal of our approach is to satisfy the Quality of Service (QoS) and reliability requirements of the system while maximizing the remaining energy of the edge device batteries. We considered a multi-gateway network and proposed a scheme with two interconnected components: *Intra-Gateway Management* and *Inter-Gateway Management*. Together, they control the offloading rates of edge devices, carry out gateway assignments, and orchestrate the routing within the network. Each of the problems are handled in a distributed fashion, resulting in a light-weight and scalable solution. The results indicate that our approach improves the lifetime of a network with 36 edge devices by 5.8 months compared to the closest approach for 2 gateways to as much as 7.4 months for 4 gateways. We also evaluated the energy savings and QoS for various network configurations. Experiments demonstrated similar energy savings compared to the state-of-the-art approaches while preserving reliability, but fewer task deadline misses.

We propose to extend this work for ML-specific applications in future work. Our computation offloading framework, handled by the *Intra-Gateway Management*, finds the best data offloading rates from an edge device to a gateway, given the workload characteristics. However, the algorithm is application-agnostic, meaning, it does not exploit the application structure. Many machine learning models have proper structures that can be broken up into sequential parts. For example, neural networks can be split to run the first few layers at the edge and the rest in the cloud. Features can be extracted at the edge and then be communicated over the internet, which significantly reduces communication costs. For future work, similar to what we proposed in this paper, we will optimize computation offloading and additionally find the optimal machine learning workload distribution between edge and cloud. The primary QoS metric for machine learning applications is the learning accuracy. Therefore, we propose to optimize the model accuracy high while minimizing training costs.

Acknowledgements

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

References

[1] International Data Corporation, "IoT Growth Demands Rethink of Long-Term Storage Strategies," https://www.idc.com/getdoc.jsp?containerId=prAP46737220, 2020, [Online].

[2] ——, "The Growth in Connected IoT Devices," https://www.idc.com/getdoc.jsp?containerId=prUS45213219, 2019, [Online].

[3] Cisco Jasper, "The hidden costs of delivering iiot services: Industrial monitoring & heavy equipment," 2016. [Online]. Available: https://www.cisco.com/c/dam/m/en_ca/never-better/manufacture/pdfs/hidden-costs-of-delivering-iiot-services-white-paper.pdf

[4] J. W. McPherson, "Reliability challenges for 45nm and beyond," in *2006 43rd ACM/IEEE design automation conference*. IEEE, 2006.

[5] T. S. Rosing, K. Mihic, and G. De Micheli, "Power and reliability management of socs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 391–403, 2007.

[6] C. Zhuo, D. Sylvester, and D. Blaauw, "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.

[7] B. Zhang, N. Mor, J. Kolb, D. S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, and J. Kubiatowicz, "The cloud is not enough: Saving iot from the cloud," in *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.

[8] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, 2017.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, 2016.

[10] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The Case for Lifetime Reliability-Aware Microprocessors," in *ACM SIGARCH Computer Architecture News*, vol. 32, no. 2, 2004.

[11] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power iot edge devices," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2016.

[12] K. Cao, G. Xu, J. Zhou, T. Wei, M. Chen, and S. Hu, "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.

[13] F. Samie, V. Tsoutsouras, D. Masouros, L. Bauer, D. Soudris, and J. Henkel, "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.

[14] W. Cui, Y. Kim, and T. S. Rosing, "Cross-Platform Machine Learning Characterization for Task Allocation in IoT Ecosystems," in *Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017.

[15] Z. Sheng, C. Mahapatra, V. C. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Transactions on Cloud Computing*, vol. 6, 2015.

[16] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *Journal on Selected Areas in Communications*, 2015.

[17] J. Henkel, S. Pagani, H. Amrouch, L. Bauer, and F. Samie, "Ultra-low power and dependability for iot devices (invited paper)," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

[18] L. Li, S. Li, and S. Zhao, "Qos-aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, 2014.

[19] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *Internet of Things Journal*, 2018.

[20] R. Li, Z. Zhou, X. Chen, and Q. Ling, "Resource price-aware offloading for edge-cloud collaboration: A two-timescale online control approach," *IEEE Transactions on Cloud Computing*, 2019.

[21] D. J. Pagliari, R. Chiaro, C. Yukai, V. Sara, M. Enrico, and P. Massimo, "Input-dependent edge-cloud mapping of recurrent neural networks inference," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[22] S. Sardellitti, G. Scutari, and S. Barbarossa, "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.

[23] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *International Symposium on Service-Oriented System Engineering*. IEEE, 2013.

[24] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.

[25] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE internet of things journal*, 2016.

[26] F. Samie, V. Tsoutsouras, S. Xydis, L. Bauer, D. Soudris, and J. Henkel, "Distributed qos management for internet of things under resource constraints," in *International Conference on Hardware/Software Codesign and System Synthesis*, 2016.

[27] M. Katsaragakis, D. Masouros, V. Tsoutsouras, F. Samie, L. Bauer, J. Henkel, and D. Soudris, "Dmrm: Distributed market-based resource management of edge computing systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019.

[28] F. Samie, L. Bauer, and J. Henkel, "Hierarchical classification for constrained iot devices: A case study on human activity recognition," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8287–8295, 2020.

[29] K. Ergun, R. Ayoub, P. Mercati, D. Liu, and T. Rosing, "Energy and qos-aware dynamic reliability management of iot edge computing systems,"

in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2021.

[30] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Distributed trade-based edge device management in multi-gateway iot," *ACM Transactions on Cyber-Physical Systems*, 2018.

[31] S. Baskar and V. Dhulipala, "Comparative analysis on fault tolerant techniques for memory cells in wireless sensor devices," *Asian Journal of Research in Social Sciences and Humanities*, vol. 6, no. cs1, pp. 519–528, 2016.

[32] J. Yao and N. Ansari, "Fog resource provisioning in reliability-aware iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8262–8269, 2019.

[33] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, 2009.

[34] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on networking*, 2004.

[35] Y. Peng, W. Qiao, L. Qu, and J. Wang, "Sensor fault detection and isolation for a wireless sensor network-based remote wind turbine condition monitoring system," *IEEE Transactions on Industry Applications*, vol. 54, no. 2, pp. 1072–1079, 2017.

[36] K. Okafor, "Dynamic reliability modeling of cyber-physical edge computing network," *International Journal of Computers and Applications*, vol. 43, no. 7, pp. 612–622, 2021.

[37] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability Modeling and Management in Dynamic Microprocessor-Based Systems," in *Design Automation Conference*, 2006.

[38] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature Aware Task Scheduling in MPSoCs," in *2007 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2007, pp. 1–6.

[39] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. Š. Rosing, "Warm: Workload-Aware Reliability Management in Linux/Android," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, 2016.

[40] L. Huang, F. Yuan, and Q. Xu, "Lifetime reliability-aware task allocation and scheduling for mpsoc platforms," in *Design, Automation & Test in Europe Conference & Exhibition*, 2009.

[41] H. Wang, L. Hu, X. Guo, Y. Nie, and H. Tang, "Compact piecewise linear model based temperature control of multi-core systems considering leakage power," *IEEE Transactions on Industrial Informatics*, 2019.

[42] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 960–965.

[43] M. Abo-Zahhad, M. Farrag, A. Ali, and O. Amin, "An energy consumption model for wireless sensor networks," in *International Conference on Energy Aware Computing Systems & Applications*. IEEE, 2015.

[44] M. R. Jongerden and B. R. Haverkort, "Which battery model to use?" *IET software*, vol. 3, no. 6, pp. 445–457, 2009.

[45] L. M. Rodrigues, C. Montez, R. Moraes, P. Portugal, and F. Vasques, "A temperature-dependent battery model for wireless sensor networks," *Sensors*, vol. 17, no. 2, 2017.

[46] C. G. Cassandras, T. Wang, and S. Pourazarm, "Optimal routing and energy allocation for lifetime maximization of wireless sensor networks with nonideal batteries," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 86–98, March 2014.

[47] F. Beneventi, A. Bartolini, A. Tilli, and L. Benini, "An Effective Gray-Box Identification Procedure for Multicore Thermal Modeling," *IEEE Transactions on Computers*, vol. 63, no. 5, 2012.

[48] P. Van Overschee and B. De Moor, "N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, 1994.

[49] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare internet of things: A case study on ecg feature extraction," in *International conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*. IEEE, 2015.

[50] A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing, "Hierarchical and distributed machine learning inference beyond the edge," in *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, 2019.

[51] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017.

[52] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[53] B. Heintz, A. Chandra, R. K. Sitaraman, and J. Weissman, "End-to-end optimization for geo-distributed mapreduce," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 293–306, 2014.

[54] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of aggregated iot traffic and its application to an iot cloud," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 679–694, 2019.

[55] L. Kleinrock, *Queueing systems, volume 2: Computer applications*. Wiley, 1976.

[56] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *INFOCOM*. IEEE, 2011.

[57] O. Iova, F. Theoleyre, and T. Noel, "Using multiparent routing in rpl to increase the stability and the lifetime of the network," *Ad Hoc Networks*, 2015.

[58] H. Sharma, A. Haque, and Z. A. Jaffery, "Maximization of wireless sensor network lifetime using solar energy harvesting for smart agriculture monitoring," *Ad Hoc Networks*, vol. 94, p. 101966, 2019.

[59] G. Torrisi, "Low-complexity numerical methods for nonlinear model predictive control," Ph.D. dissertation, ETH Zurich, 2017.

[60] J. Lunze, *Feedback control of large scale systems*. Prentice Hall, 1992.

[61] G. F. Franklin, J. D. Powell, M. L. Workman *et al.*, *Digital control of dynamic systems*. Addison-wesley Reading, MA, 1998, vol. 3.

[62] J. Mazzola and A. Neebe, "Bottleneck generalized assignment problems," *Engineering Costs and Production Economics*, 1988.

[63] J. B. Mazzola and A. W. Neebe, "An algorithm for the bottleneck generalized assignment problem," *Computers & operations research*, 1993.

[64] S. Martello and P. Toth, "The bottleneck generalized assignment problem," *European journal of operational research*, 1995.

[65] Y. Fu, J. Sun, K. Lai, and J. W. Leung, "A robust optimization solution to bottleneck generalized assignment problem under uncertainty," *Annals of Operations Research*, vol. 233, no. 1, pp. 123–133, 2015.

[66] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on wireless communications*, vol. 5, no. 8, pp. 2185–2193, 2006.

[67] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[68] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2003.

[69] MagPi, "The official Raspberry Pi Magazine," https://magpi.raspberrypi.org/, 2019, [Online].

[70] Kaggle, "Historical Hourly Weather Data," https://www.kaggle.com/selfishgene/historical-hourly-weather-data, 2017, [Online].

[71] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers*. IEEE, 2012.

[72] X. Yu, K. Ergun, L. Cherkasova, and T. Š. Rosing, "Optimizing sensor deployment and maintenance costs for large-scale environmental monitoring," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3918–3930, 2020.

[73] "ODYS QP Solver," https://www.odys.it/qp-solver-for-embedded-optimization/, [Online].

[74] V. Valls, G. Iosifidis, and T. Salonidis, "Maximum lifetime analytics in iot networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1369–1377.

[75] T. P. Raptis, A. Passarella, and M. Conti, "Maximizing industrial iot network lifetime under latency constraints through edge data distribution," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 708–713.

[76] I. Silva, R. Leandro, D. Macedo, and L. A. Guedes, "A dependability evaluation tool for the internet of things," *Computers & Electrical Engineering*, vol. 39, no. 7, pp. 2005–2018, 2013.

[77] A. Shrestha, L. Xing, Y. Sun, and V. M. Vokkarane, "Infrastructure communication reliability of wireless sensor networks considering common-cause failures," *International Journal of Performability Engineering*, vol. 8, no. 2, p. 141, 2012.

[78] S. Sinche, O. Polo, D. Raposo, M. Femandes, F. Boavida, A. Rodrigues, V. Pereira, and J. S. Silva, "Assessing redundancy models for iot reliability," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 14–15.

[79] H. M. AboElFotoh, S. S. Iyengar, and K. Chakrabarty, "Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures," *IEEE transactions on reliability*, vol. 54, no. 1, pp. 145–155, 2005.

[80] D. Macedo, L. A. Guedes, and I. Silva, "A dependability evaluation for internet of things incorporating redundancy aspects," in *Proceedings of the 11th IEEE international conference on networking, sensing and control*. IEEE, 2014, pp. 417–422.

[81] D. Bruneo, A. Puliafito, and M. Scarpa, "Energy control in dependable wireless sensor networks: a modelling perspective," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 225, no. 4, pp. 424–434, 2011.

[82] L. Xing, "Reliability in internet of things: Current status and future perspectives," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6704–6721, 2020.

[83] Z. W. Birnbaum and S. C. Saunders, "A new family of life distributions," *Journal of applied probability*, vol. 6, no. 2, pp. 319–327, 1969.

[84] M. L. Fairbairn, I. Bate, and J. A. Stankovic, "Improving the dependability of sensornets," in *2013 IEEE international conference on distributed computing in sensor systems*. IEEE, 2013, pp. 274–282.