UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Energy-Efficient and Reliability-Driven Management of IoT Systems

Permalink

https://escholarship.org/uc/item/4gz2h8r3

Author

Ergun, Kazim

Publication Date

2022

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA SAN DIEGO

Energy-Efficient and Reliability-Driven Management of IoT Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Machine Learning and Data Science)

by

Kazim Ergun

Committee in charge:

Professor Tajana Šimunić Rosing, Chair
Professor Tara Javidi
Professor Ryan Kastner
Professor Piya Pal
Professor Xinyu Zhang

2022

Copyright

Kazim Ergun, 2022

All rights reserved.

The dissertation of Kazim Ergun is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

To my parents, Sevim and Vasfi,
and my sister Ümit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank all the people who directly or indirectly supported me during my PhD. First, I want to sincerely thank my advisor, Prof. Tajana Rosing for her guidance, trust, and encouragement. Her experience and knowledge has kept me in the right direction. I am extremely grateful for her supervision in work and understanding in off-work difficulties, that have made this PhD possible. I would also like to thank my dissertation committee members, Prof. Tara Javidi, Prof. Ryan Kastner, Prof. Piya Pal, and Prof. Xinyu Zhang for their valuable feedback and discussions related to my Ph.D. work.

I have to thank Lucy Cherkasova, who was my supervisor when I was an intern at Arm. I am grateful to her for providing me such an opportunity and supporting me both during and after my internship. I also want to sincerely thank Raid Ayoub and Pietro Mercati for all the support and guidance they provided during a joint project. My research was made possible by funding from the National Science Foundation (NSF) Grant 1527034, 1730158, 1826967, 1911095, 2003279, CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA, and SRC-Global Research Collaboration grant.

I would like to thank all my colleagues in SEELab for their active collaboration, continuous support, and the good memories. It was a privilege to work with and along-side them. I am grateful to all of them, but I would also like to give special thanks to Behnam Khaleghi, Xiaofan Yu, Justin Morris, and Nitish Nagesh.

Most importantly, I owe so much to my parents, my sister, and all my family for their unconditional, unending love, understanding, and patience that helped me navigate through the difficult moments during the last years. Finally, I thank all my friends whose constant support and liveliness kept me going everyday.

Chapter 2 contains material from "*RelIoT*: Reliability Simulator for IoT Networks", by Kazim Ergun, Xiaofan Yu, Nitish Nagesh, Lucy Cherkasova, Pietro Mercati, Raid Ayoub, Tajana Rosing, which appears in International Conference on Internet of Things (ICIOT), 2020 [1]. The dissertation author was the primary investigator and author of this paper.

Chapter 3 contains material from “Dynamic Reliability Management of Multi-Gateway IoT Edge Computing Systems”, by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in IEEE Internet of Things Journal [2]. The dissertation author was the primary investigator and author of this paper.

Chapter 4 contains material from “Reliability-Aware Routing in IoT Networks Using Reinforcement Learning”, by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in Ad Hoc Networks [3]. The dissertation author was the primary investigator and author of this paper.

Chapter 5 contains material from “Dynamic Optimization of Battery Heath in IoT Networks”, by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in International Conference on Computer Design (ICCD), 2019 [4]. The dissertation author was the primary investigator and author of this paper.

VITA

2017 B. S. in Electrical and Electronics Engineering (Valedictorian), Middle East Technical University (METU), Ankara, Turkey

2022 Ph. D. in Electrical Engineering (Machine Learning and Data Science), University of California San Diego

PUBLICATIONS

Rishikanth Chandrasekaran, Kazim Ergun, Ji Hyun Lee, Dhanush Nanjunda, Tajana Rosing, "FHDnn: Communication Efficient and Robust Federated Learning for IoT Networks", *Design Automation Conference (DAC)*, 2022.

Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, "Dynamic Reliability Management of Multi-Gateway IoT Edge Computing Systems", *IEEE Internet of Things Journal*, 2022.

Emily Ekaireb, Xiaofan Yu, Kazim Ergun, Quanling Zhao, Kai Lee, Muhammad Huzaifa, Tajana Rosing, "ns3-fl: Simulating Federated Learning with ns-3", *Proceedings of the Workshop on ns-3*, 2022.

Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli and T. Rosing, "HyDREA: Utilizing Hyperdimensional Computing For A More Robust and Efficient Machine Learning System", *IEEE Transactions on Embedded Computing Systems (TECS)*, 2022.

Xiaofan Yu, Kazim Ergun, Xueyang Song, Lucy Cherkasova, Tajana Rosing, "Automating and Optimizing Reliability-Driven Deployment in Energy-Harvesting IoT Networks", *IEEE Transactions on Network and Service Management (TNSM)*, 2022.

Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, "Reliability-Aware Routing in IoT Networks Using Reinforcement Learning", *Ad Hoc Networks*, 2022.

Justin Morris, Kazim Ergun, Behnam Khaleghi, Mohsen Imani, Baris Aksanli and Tajana Rosing, "Hyperdimensional Computing: Towards More Reliable and Efficient Machine Learning Systems", *Design, Automation, and Test in Europe (DATE)*, 2021.

Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, "Improving Mean Time to Failure of IoT Networks with Reliability-Aware Routing", *10th Mediterranean Conference on Embedded Computing (MECO)*, 2021.

Kazim Ergun, Raid Ayoub, Pietro Mercati, Dan Liu, Tajana Rosing, "Energy and QoS-Aware Dynamic Reliability Management of IoT Edge Computing Systems", *26th Asia and South Pacific*

Design Automation Conference (ASP-DAC), 2021.

Xiaofan Yu, Kazim Ergun, Lucy Cherkasova, Tajana Rosing, "Optimizing Sensor Deployment and Maintenance Costs for Large-Scale Environmental Monitoring", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

Kazim Ergun, Xiaofan Yu, Nitish Nagesh, Lucy Cherkasova, Pietro Mercati, Raid Ayoub, Tajana Rosing, "Simulating Reliability of IoT Networks with *RelIoT*", *IEEE 50th Conference on Dependable Systems and Networks (DSN)*, 2020.

Kazim Ergun, Xiaofan Yu, Nitish Nagesh, Lucy Cherkasova, Pietro Mercati, Raid Ayoub, Tajana Rosing, "*RelIoT*: Reliability Simulator for IoT Networks", *International Conference on Internet of Things (ICIOT)*, 2020.

Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, "Dynamic Optimization of Battery Heath in IoT Networks", *37th International Conference on Computer Design (ICCD)*, 2019.

Berk Kaya, Mertalp Ocal, Can Ufuk Ertenli, Kazim Ergun, "Path Planning of Differential Drive Robots with Image Based Object Tracking", *19th Turkish National Conference on Automatic Control (TOK)*, 2017.

Mertalp Ocal, Kazim Ergun, Gozde Bozdagi Akar, "Comparative Analysis of Hyperspectral Feature Extraction Methods in Vegetation Classification", *IEEE Conference on Signal Processing and Communications Applications (SIU)*, 2017.

ABSTRACT OF THE DISSERTATION

Energy-Efficient and Reliability-Driven Management of IoT Systems

by

Kazim Ergun

Doctor of Philosophy in Electrical Engineering (Machine Learning and Data Science)

University of California San Diego, 2022

Professor Tajana Šimunić Rosing, Chair

The Internet of Things (IoT) integrates heterogeneous devices, ranging from sensors to smartphones, tablets and edge servers, and can provide a variety of services, beyond the traditional Internet. Unfortunately, due to its unprecedented scale and ubiquity, IoT faces a maintainability challenge and a set of interrelated problems. With the emergence of edge computing, IoT devices execute various tasks that consume a significant amount of power to deliver high quality of service, which can drain their battery in short time. High peak power increases the device temperature stress, which worsens the impact of transistors and interconnects reliability degradation mechanisms. Such mechanisms lead to early device failures and are costly to fix. In this dissertation, we focus on novel solutions for energy-efficient and

reliability-driven management of IoT systems. We introduce a simulation framework called *RelIoT* to enable reliability evaluation and analysis in IoT networks, which paves the way for the development of new network management solutions. We develop a dynamic reliability management technique based on computation offloading for IoT edge computing architectures. Our approach achieves 20.5% longer mean time to failure than the next best network management solution. We also present an adaptive and distributed reliability-aware routing protocol using reinforcement learning. We show that our routing protocol can improve reliability of a network up to 73.2% compared to state-of-the-art routing approaches. The main focus in all our solutions is to use device batteries efficiently, satisfy QoS requirements, and improve overall network lifetime by mitigating reliability degradation. Lastly, we complement this to specifically study battery health and associated degradation mechanisms, as the traditional techniques developed for optimizing the energy consumption of networks do not yield optimal battery life. An improvement in network lifetime up to 68.5% can be achieved with our approach compared to energy consumption optimization approaches.

Chapter 1

Introduction

The Internet of Things (IoT) continues to grow as it is adopted progressively across many domains such as logistics, farming, industrial and environmental monitoring, healthcare, and smart infrastructures [5]. The rapid development of IoT has lead to a tremendous increase in the number of devices connected to the Internet. There are already more than 10 billion interconnected heterogeneous devices, ranging from low-power sensors with limited computational capabilities to multi-core platforms on the high-end [6]. This number is expected to exceed 40 billion by 2025, requiring a sophisticated and agile network hierarchy, along with more complex management. Another challenge that stems from such dramatic growth is the maintainability of IoT systems. Despite the wide range of different IoT devices, all electronics age and degrade, eventually leading to failures that necessitate costly maintenance. Currently, operational expenses account for 80% of the overall spending [7] ($746 billion in 2019 [8]), of which a significant fraction is associated with maintenance and technical diagnostics due to system failures. While meeting the needs of an evolving range of applications, it is also a crucial requirement for IoT devices and networks to operate reliably for long periods. Otherwise, maintenance investments can become a critical bottleneck for the growth of IoT.

The ultimate goal of IoT systems is to deliver a high quality of service to users, where the service can be data acquisition and communication, information processing, or decision making [9]. The level of users' satisfaction, described by Quality of Service (QoS) metrics such

as delay, throughput, accuracy, improves with higher system performance. Improvements in system performance can be achieved through increased computation or communication, which result in higher power draw and reduced battery life. Power dissipation also raises the device temperature quickly due to the absence of active cooling solutions. High temperature stress, in turn, accelerates reliability degradation [10]. Previous research has shown that reliability degradation of electronics worsens exponentially with increasing temperature because of the intensified effects of various mechanisms such as Time-Dependent Dielectric Breakdown (TDDB) and Bias Temperature Instability (BTI) [11, 12]. As a result of these mechanisms, components in IoT devices age and degrade, leading to higher rates of early failures.

The reliability of IoT networks can be improved with proper system design and reliability management strategies. Prior efforts have focused primarily on developing useful applications, improving the design and management of performance, energy, and reliability trade-offs for individual devices [13]. There is a large body of studies investigating the dynamic management of reliability degradation for processor-based systems, showing notable gains in the lifetime of stand-alone devices such as desktop and server machines, mobile devices, etc. [10, 14, 12]. However, at the level of networks of devices, the focus is mainly on improving performance with energy-efficient solutions and achieving extended battery life [15]. Unfortunately, very little work has been done to consider a network-level IoT reliability management framework. The approaches proposed for individual devices may not yield optimal strategies for groups of devices and sometimes cannot be directly applied to networked systems. In IoT networks, a group of interconnected devices cooperate to accomplish a common task. Hence, there are interactions between devices, data dependencies, and a large optimization space, making the management complex.

In this dissertation, we propose the design of an exploration, simulation, and optimization framework for IoT systems, develop end-to-end management solutions capable of evaluating device and network level tradeoffs with respect to performance (QoS requirements), power consumption (battery lifetime), thermal issues (including the effects of environmental conditions

on devices), and the resulting reliability and maintenance costs. We define four interconnected challenges described below that we address to obtain an end-to-end framework.

Simulation and Exploration of IoT Systems

No common infrastructure has been developed that allows engineers and researchers to simulate, explore, and optimize IoT systems from not just performance and energy consumption perspectives, but also with the view of ensuring good system-wide reliability and maintainability. The lack of such tools has prevented researchers from undertaking reliability-oriented analysis, exploration, and predictions early in the design cycle. A proper tool can lead to the design of new network reliability management strategies.

Computation and Communication Optimization

IoT devices are responsible for the collection, aggregation, and processing of data. In addition to this computational load, they have to communicate large amounts of data, either for distributed processing across other network devices, or for in-depth analysis, long-term storage, and permanent archiving in the cloud. Therefore, both computation and communication play a role for the performance of an IoT service. At the same time, the power dissipation, along with the resulting temperature and reliability degradation of the device, varies as a function of computation and computation loads. It is of great importance to include the contributions of both components in reliability analysis and optimize accordingly, considering their trade-offs for different applications.

Network-Level Management

In addition to leveraging the reliability management techniques developed for individual, standalone devices, an IoT device can benefit from network-level strategies, such as computation offloading, resource allocation, or load balancing [16]. However, this adds another layer of complexity to the problem at hand. Individual device controls include voltage and frequency levels, duty-cycling, sampling rates and so on, while network-level controls dictate the interactions between devices, i.e., communication rates and paths, distributed workloads and others. A successful management system that looks over a large number of network devices needs to be

dynamic, scalable with low complexity, and ideally have distributed operation.

Modular Management

One should not think of IoT devices as single "black box" entities if fine-grained management is desired. Rather, a better model is to divide them into components: sensing unit, processing unit, RF transceiver, power unit (battery), and energy harvester. To obtain the overall characterization of an IoT device, the power, thermal, and reliability contributions of different components should be combined.

This thesis first introduces a simulation framework called *RelIoT* to enable reliability evaluation and analysis in IoT networks, which paves the way for the development of new network management solutions. It then analyzes different network configurations and explores their energy-reliability trade-offs, all of which are simulated using the proposed framework and validated on real testbeds. We continue with a dynamic reliability management technique based on computation offloading for IoT edge computing architectures. We also present an adaptive and distributed reliability-aware routing protocol using reinforcement learning. The main focus in our solutions is to use device batteries efficiently, satisfy QoS requirements, and improve overall network lifetime by mitigating reliability degradation. Lastly, we specifically study battery health and associated degradation mechanisms, as the traditional techniques developed for optimizing the energy consumption of networks do not yield optimal battery life.

Next, we discuss the key problems addressed by this dissertation; reliability modeling and simulation, dynamic reliability management, reliability-aware routing, and battery health management. We highlight the main challenges in the context of IoT networks, then present our solutions and the achieved results.

1.1 Simulating Reliability of IoT Networks

Despite being a major concern, there is a lack of thorough studies on the reliability of IoT networks in the context of device aging and degradation. Reliability-aware design and

management were shown to delay failures and improve the lifetime of individual devices [12, 10]. Even though IoT networks can also radically benefit from this, the unavailability of tools that provide reliability modeling for networks makes it difficult to assess reliability-aware strategies. To extend the work on individual devices and explore different management strategies at a network-level, there is a need for a convenient tool.

Simulators are widely used tools in research and industry to evaluate and validate networks, and to study novel methods without the need for real deployments when resources are limited. However, currently, it is not possible to analyze the reliability of IoT networks with any of the available network simulators. The most popular ones, e.g., ns-3 [17], OMNeT++ [18], and OPNET [19], are designed only for analyzing communication performance (throughput, delay, utilization, etc.) under different protocols, and lack the ability to account for computation power, performance, and reliability of network devices.

Motivated by this gap in existing simulation platforms, there has been growing attention towards incorporating some of these considerations into simulation tools. While there has been some attention paid to these considerations individually, no prior work has addressed these concerns simultaneously and using a sufficiently accurate modeling strategy. For instance, due to the energy constraints in battery-powered sensor networks, several works have integrated power modeling and analysis with different granularity. Wu *et al.* [20] first introduced energy source models and device energy models to ns-3. They use existing analytical battery models and build communication energy models based on radio hardware datasheets. In another work named PASES [21], the authors construct accurate power consumption models for both processor and radio components of network devices by hardware design space exploration. In network simulators, usually, the accuracy of power models is compromised for attaining low computational costs. To provide flexible options for heterogeneous devices in an IoT network, the simulator should offer configurable models with different granularity, while allowing extensions for user-specified models.

For IoT performance, iFogSim [22] and EdgeCloudSim [23] incorporated computation

performance (e.g. processing delay) to simulate end-to-end latency of a multi-level IoT structure including cloud servers, gateways, and sensors. Both toolkits employ low accuracy estimations such as look-up tables for latency and power analysis. Although such models are favorable in terms of computational overhead, it is important to capture the characteristics of different types of IoT devices running various applications, by doing a finer-grained estimation. Besides power and performance models, the simulator should also include modeling of temperature, all required to work together for evaluating reliability.

In Chapter 2, we take the first step in presenting an integrated reliability framework for IoT networks based on the ns-3 simulator. The framework, in addition to reliability, incorporates three other interrelated models, namely power, performance, and temperature, each of having a direct influence on reliability. We validate our framework on a mesh network testbed. Finally, we show how the framework offers an opportunity for researchers to explore and optimize trade-offs between energy, performance and reliability in IoT networks.

1.2 Dynamic Reliability Management of IoT Edge Computing Systems

IoT brings significant demands for data collection, storage, and processing [24]. In traditional cloud-centric IoT, simple tasks such as data collection, aggregation, and preliminary processing are done at the edge, whereas more resource intensive workloads like in-depth analysis and long-term storage are carried out in the cloud. As a result, data needs to be constantly moved from the edge to cloud. Naively communicating data at large volumes from the devices to the cloud leads to expensive infrastructure and service needs, high energy costs, and causes unreliable latency.

The emerging paradigm of edge computing envisions overcoming the shortcomings of the cloud-centric Internet of Things (IoT) by providing data processing and storage capabilities closer to the source of data [25]. By 2025, around 40% of IoT-generated data will be processed,

stored, and acted upon close to the edge of network [26]. Accordingly, with the increasing demand of computation workloads on them, IoT edge devices are prone to failures more than ever. Intense workload requirements raise the average power consumption along with the device temperature. This dramatically exacerbates the impact of temperature-dependent transistor and interconnect reliability degradation mechanisms that cause semiconductor devices to fail. CMOS transistors, which are used in almost all the electronic devices, are affected by Time Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI). Also, metal interconnects in electronics are affected by Electromigration (EM) and Thermal Cycling (TC) [27]. The continuous scaling of CMOS transistor and interconnect dimensions intensifies the impact of degradation, contributing to early device failures.

The International Technology Roadmap for Semiconductors (ITRS) identifies reliability issues as one of the primary concerns for integrated circuits [28]. Uncertainties in reliability can lead to performance penalties and field failures that are costly to fix as well as damaging to the manufacturer reputation. Therefore, researchers have come up with various techniques at different levels of development hierarchy (process [29], design, OS [10], software [30], and application [31]) to cope with reliability degradation [32]. All the degradation mechanisms can be described by a reliability function which represents the probability that the device does not fail at any point in time [33]. Then, the degradation of devices can be controlled at runtime by adjusting operation dynamics that influence temperature. Such strategy is called Dynamic Reliability Management (DRM) and is usually implemented to meet a lifetime criteria for the target device. In DRM, degradation is tracked with model-based estimation [12] or monitored using on chip or on board sensors [34, 35].

DRM has been researched and used extensively for personal computers, mobile devices, and data centers [36, 10, 37]. However, our work was the first to consider a network-level IoT reliability management framework [1]. In addition to leveraging the DRM techniques developed for individual and stand-alone devices, an IoT device can benefit from network-level mitigation strategies, such as offloading computation or allocating network resources,

for alleviating reliability degradation. Several prior works [38, 39, 40] 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.

For typical prior work on edge computing systems [41, 42, 38, 43], 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 [44]. Many studies aim at balancing the tradeoff between power consumption and delay performance [41, 45, 46, 42, 47]. The problem of QoS management for IoT edge devices under the limited resources (bandwidth, gateway's processing power) of edge computing systems was addressed in [48, 39, 49].

In Chapter 3, 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.

1.3 Reliability-Aware Routing

The term reliability, in network related studies, 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. The so-called node errors can be categorized into three groups: soft errors (causing random bit flips) [50], software reliability issues [51], or batteries running out of energy [52, 53], none of which handle hard errors. For example, [51] considers software failures, message congestion, VM failures on IoT devices. 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 [54]), 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.

Hard failures caused by well-known thermally-driven mechanisms in silicon, such as TDDB, EM, BTI, result in a need to replace that electronic component in the field, leading to high maintenance and replacement costs. Hard failure models have been studied extensively at the circuit and chip level [11, 55, 12], and adopted for dynamic voltage & frequency scaling, task scheduling, and power gating strategies in multi-core system-on-a-chips [10]. Prior studies have not considered the impact of hard failures at the network level and how networking might affect the electronics reliability. IoT devices are often deployed in harsh environments, causing stress on the hardware that reduces their reliability and mean time to failure (MTTF). Despite temperature being a key driver in the rate of failure, the majority of IoT devices do not have active cooling to mitigate the thermal stress. In such cases, curbing power dissipation of devices helps in lowering the device temperatures and scaling down the effect of temperature-driven failure mechanisms to achieve a better MTTF. Network routing can be useful in this regard; it is possible to place the IoT devices into low-power states by avoiding them in communication paths. In this way, low reliability devices in the network are utilized less to reduce thermal stress and slow down degradation.

The literature on network routing does not scrutinize the problem of hardware failures and reliability issues as a bottleneck for the lifetime of networks. Since the majority of IoT devices are battery-operated, the works in this domain are directed towards improving battery lifetimes [52, 56, 57]. Maximum lifetime routing incorporates the residual energy of device batteries into routing decisions and ensures a balanced depletion among them. The typical approach for routing is to model the network as a weighted directed graph and then find paths with the minimum cumulative weight. The weights of the graph edges and vertices traditionally include a variety of node and link metrics: latency, hop count, stability, bandwidth, throughput, and energy, or might instead, be a composite of multiple metrics [58, 59]. The impact of degradation mechanisms on device reliability has not been taken into account by routing techniques to date.

In Chapter 3, we present a novel adaptive and distributed reliability-aware routing protocol based on reinforcement learning to mitigate the reliability degradation of IoT devices and improve the network MTTF. The proposed algorithm adapts routing decisions based on the current reliability status of the devices, the amount of degradation they are likely to experience due to communication activity, and network performance goals. We extend the ns-3 [60] network simulator to support our reliability models and evaluate the routing performance by comparing with state-of-the-art approaches.

1.4 Dynamic Optimization of Battery Health

Another key driver of network failure is when an IoT device shuts off due to lack of power. Replacing batteries for devices across a network involves expensive labor and infrastructure, hence the battery life should be improved to keep the network running for as long as possible. There is often confusion when discussing battery lifetime because the lifetime for rechargeable and non-rechargeable batteries are described in different ways. Non-rechargeable batteries die and need to be replaced after their initial charge is completely depleted. Therefore, the indicator for remaining battery life is the State of Charge (SoC). On the other hand, rechargeable batteries can withstand hundreds of charge-recharge cycles, allowing operation for extended periods when combined with energy harvesting solutions such as solar cells or thermal energy. Despite their ability to be recharged, these batteries still have limited lifetimes and reduced battery capacity due to battery aging. In this case, instead of SoC, we need to consider their State of Health (SoH) which is a figure of merit of the physical condition of a battery. SoH degrades due to cycle aging (charge-discharge rate and total number of cycles) and calendar aging (ambient conditions, e.g. temperature) which result in deterioration of battery conditions in the form of internal impedance increase, open voltage decrease, and most importantly, capacity fading. Depending on its application, a rechargeable battery reaches its end of life with an SoH between 70%-80% and needs to be replaced [61].

Most works in the literature concerning lifetime maximization either consider non-rechargeable batteries and deal with SoC, or assume ideal (nondegradable) operation for rechargeable batteries, neglecting the effects of SoH degradation in their management strategies. The publications that study non-rechargeable batteries try to maximize the time at which the batteries drain out of energy [62],[63]. Recent studies [64],[65] involve battery dynamics that are able to capture the "non-ideal" (nonlinear and dynamic) behavior of actual batteries in their optimization formulations. Even though they show that one can achieve a significantly longer lifetime with an optimal routing policy using a non-ideal battery model, the proposed solution does not suit systems with rechargeable batteries.

Another set of publications investigate energy harvesting networks with rechargeable batteries. This work usually tries to develop control algorithms to optimally utilize available energy [66],[67]. However, only a handful of studies [68, 69, 70] consider the degradation of batteries, which is the major factor in determining the lifetime of a network of devices with rechargeable batteries.

The degradation of batteries in a network control problem is studied primarily in battery energy storage systems [71, 72], smart grid [73], and data centers [74, 75]. These works are not directly applicable to IoT domain because of the different structures of the network, and additional constraints that the network possesses. In energy storage systems and smart grid, batteries are often modeled in an aggregate fashion. In IoT networks, the batteries from different devices are not physically connected and can only supply energy to the associated device. The devices work together to accomplish a network-level task, but their energy demands are individual, which differentiates other domains from IoT.

In Chapter 4, we formulate the problem of determining the data flow that minimizes SoH degradation of rechargeable batteries for an IoT network, then propose a model predictive controller (MPC) solution. We incorporate battery models which capture the temperature-dependent, nonlinear charging/discharging and degradation behavior into the system model. We evaluate our solution using real-world deployment in a smart home and a large scale IoT network

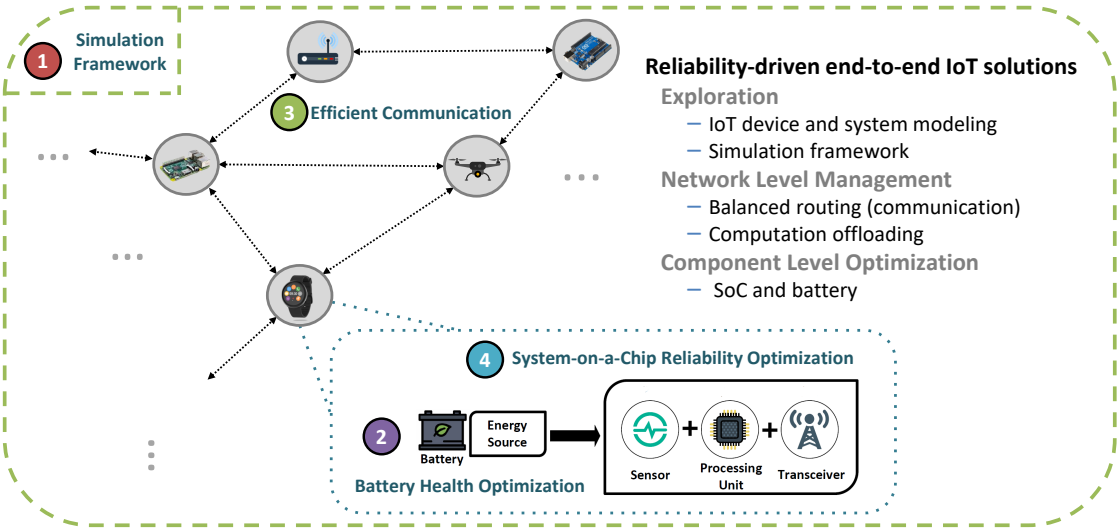


Figure 1.1. Thesis contributions: reliability-driven, energy-efficient end-to-end IoT solutions.

HPWREN [76].

1.5 Thesis Contributions

This thesis focuses on energy-efficient and reliability-driven end-to-end solutions for IoT systems. It presents methods to improve the lifetime of IoT networks using reliability-aware approaches. It covers simulation, exploration, management, and optimization aspects to provide a comprehensive treatment. A reference schema for the scope of this dissertation is presented in Figure 1.1. The following discussion demonstrates the contributions and the outlines of the rest of the thesis:

- It presents an integrated reliability framework for IoT networks based on the ns-3 simulator. The lack of such tools has restrained researchers from performing reliability-oriented analysis, exploration, and early predictions in the design cycle. Our contribution facilitates the aforementioned processes, which can lead to the design of new network reliability management strategies. The proposed framework, in addition to reliability, incorporates three other interrelated models - power, performance, and temperature - which are required

to model reliability. We validate our framework on a mesh network with ten heterogeneous devices of three different types. We demonstrate that the models accurately capture the power, temperature, and reliability dynamics of real networks. We finally simulate and analyze two examples of energy-optimized and reliability-optimized network configurations to show how the framework offers an opportunity for researchers to explore trade-offs between energy and reliability in IoT networks. The simulation framework is described in Chapter 2.

- It proposes a novel dynamic reliability management (DRM) technique for multigateway 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 on 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-theart approaches while satisfying reliability requirements, where other approaches fail by a large margin. This work is presented in Chapter 3.

- It proposes a novel adaptive and distributed reliability-aware routing protocol based on reinforcement learning to mitigate the reliability degradation of IoT devices and improve the network Mean Time to Failure (MTTF). Through routing, we curb the utilization of quickly degrading devices, which helps to lower the device power dissipation and

temperature, thus reducing the effect of temperature-driven failure mechanisms. To quantify and optimize networking performance besides reliability, we incorporate Expected Transmission Count (ETX) in our formulations as a measure of communication link quality. Our proposed algorithm adapts routing decisions based on the current reliability status of the devices, the amount of degradation they are likely to experience due to communication activity, and network performance goals. We extend the ns-3 network simulator to support our reliability models and evaluate the routing performance by comparing with state-of-the-art approaches. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance. The details of this routing method are described in Chapter 4.

- It formulates the problem of minimizing battery degradation to improve the lifetime of IoT networks and solve it with Model Predictive Control (MPC) leveraging models for battery dynamics and State of Health (SoH). The battery SoH is modeled using a realistic non-linear model while taking ambient temperature into account. We demonstrate that our solution can improve network lifetime up to 68.5% compared to the conventional energy consumption focused algorithms, which use simple linear battery models. The proposed approach achieves near-optimal performance in terms of preserving battery health, staying within 8.7% SoH with respect to an ideal oracle solution on average. This work is presented in Chapter 5.

Chapter 2

RelIoT: Reliability Simulator for IoT Networks

The next era of the Internet of Things (IoT) calls for a large-scale deployment of edge devices to meet the growing demands of applications such as smart cities, smart grids, and environmental monitoring. From low-power sensors to multi-core platforms, IoT devices are prone to failures due to the reliability degradation of electronic circuits, batteries, and other components. As the network of heterogeneous devices expands, maintenance costs due to system failures become unmanageable, making reliability a major concern. Prior work has shown the importance of automated reliability management for meeting lifetime goals for individual devices. However, state-of-the-art network simulators do not provide reliability modeling capabilities for IoT networks.

In this chapter, we present an integrated reliability framework for IoT networks based on the ns-3 simulator. The lack of such tools restrained researchers from doing reliability-oriented analysis, exploration, and predictions early in the design cycle. Our contribution facilitates this, which can lead to the design of new network reliability management strategies. The proposed framework, besides reliability, incorporates three other interrelated models - power, performance, and temperature - which are required to model reliability. We validate our framework on a mesh network with ten heterogeneous devices, of three different types. We demonstrate that the models accurately capture the power, temperature, and reliability dynamics of real networks.

We finally simulate and analyze two examples of energy-optimized and reliability-optimized network configurations to show how the framework offers an opportunity for researchers to explore trade-offs between energy and reliability in IoT networks.

2.1 Introduction

The Internet of Things (IoT) is a growing network of heterogeneous devices, combining residential, commercial, and industrial domains. Devices range from low-power sensors with limited computational capabilities to multi-core platforms on the high-end. From small scale (e.g., smart homes) to large scale (e.g., smart cities) applications, the IoT provides infrastructure and services for enhancing the quality of life and use of resources. By 2025, the IoT is expected to connect 41 billion devices [8].

The unprecedented scale and heterogeneity of the IoT pose major research challenges that have not been faced before. While ongoing research efforts aim at optimizing power efficiency and performance [77, 78], an aspect that has often been neglected is the reliability of the devices in the network. The common property for these devices is that they age, degrade and eventually require maintenance in the form of repair, component replacement, or complete device replacement. Since an enormous number of heterogeneous devices are interconnected in IoT networks, the maintenance costs will increase accordingly. Cisco recently anticipated that for 100K devices that operate IoT smart homes, around $6.7M/year will be spent for administration and technical diagnosis related to system failures, comprising between 30% to 70% of total costs [7]. Without a proper reliability management strategy, IoT solutions are strongly limited as it becomes infeasible to maintain increasingly large networks.

Exploring reliability management strategies requires a convenient tool for reliability evaluation. In respect of this, simulators are widely used tools in research and industry to evaluate and validate networks, to study novel methods without the need for real deployments when resources are limited. However, existing network simulators do not support aging/degradation and

reliability modeling or analysis. Popular network simulators, e.g., ns-3 [17], OMNeT++ [18], and OPNET [19] are equipped with a rich collection of communication models, allowing the assessment of network performance (throughput, delay, utilization, etc.) under different protocols. Recent research also integrated energy models and an energy-harvesting framework to the platform [20, 79]. Yet it is not possible to analyze the reliability of IoT networks with the existing tools because they lack built-in reliability models. It should be noted that we refer to the aging and reliability degradation of the hardware of an IoT device, and not to the communication reliability.

To address this gap in reliability analysis, we propose a simulation framework named *RelIoT*, which allows practical and large-scale reliability evaluation of IoT networks. The framework is implemented in ns-3 [17], a discrete-event network simulator with low computational overhead and low memory demands. Up to one billion nodes can be simulated with ns-3 [80]. In recent years, with the addition of models for various network settings and protocols through open-source contributions, ns-3 has established itself as a de facto standard network simulation tool. To allow reliability simulation in ns-3, *RelIoT* integrates the following modules:

- *Power Module:* Supports power consumption simulation for various workloads with configurable power models.
- *Performance Module:* Works in cooperation with the power module to provide performance predictions for a given workload.
- *Temperature Module:* Estimates the internal temperature of a device based on its power consumption.
- *Reliability Module:* Evaluates the device reliability using the existing thermal-based degradation models [33, 14, 12].

To the best of our knowledge, *RelIoT* is the first reliability analysis framework for heterogeneous IoT networks, taking thermal characteristics as well as power and performance

into account. *RelIoT* enables researchers to explore trade-offs between power, performance, and reliability of network devices. Moreover, *RelIoT* incurs only a marginal performance overhead on the default ns-3, making it scalable for simulating large networks. For the scalability analysis of the default ns-3, the reader is referred to previous works [80, 81]. We validate our framework with two real-world experiments, showing *RelIoT* estimates power, performance, and temperature with errors of less than 3.8%, 4.5%, and $\pm 1.5°C$ respectively. We validate reliability models against the results from existing literature. Finally, we built a mesh network testbed to illustrate that *RelIoT* can effectively capture the average long-term power and thermal behavior of devices in a dynamic network. Finally, we provide example simulation results from *RelIoT* to motivate the need for reliability-aware management and to show the differences between energy-driven and reliability-driven management strategies.

The rest of the chapter is organized as follows: Section 2.2 reviews power, performance, and reliability simulation techniques introduced by previous works. The overall structure of our proposed framework and details of models is elaborated in Section 2.3 and Section 2.4. Section 2.5 describes the evaluation setup and further discusses the results.

2.2 Related Work

Network Simulators for Power and Performance.

Network simulators are used to study the behavior of computer networks and evaluate communication protocols prior to deployment. Popular examples are: ns-3 [17], OMNeT++ [18], and OPNET [19], all of which are discrete event-based and open-source. The standard versions of these network simulators are designed only for analyzing communication performance, lacking consideration for computation power, performance, and reliability of network devices.

Motivated by energy constraints in battery-powered sensor networks, several works have integrated power modeling and analysis with different granularity. Wu *et al.* [20] first introduced energy source models and device energy models to ns-3. They used existing analytical

battery models and relied on hardware datasheets to build WiFi radio energy models. In another work named PASES [21], the authors construct accurate power consumption models for both processor and radio components of network devices by hardware design space exploration. In network simulators, usually, the accuracy of power models is compromised for attaining low computational costs. To provide flexible options for heterogeneous devices in an IoT network, *RelIoT* offers two configurable power models with different granularity, while allowing extensions for user-specified models.

For IoT performance, iFogSim [22] and EdgeCloudSim [23] incorporated computation performance (e.g. processing delay) to simulate end-to-end latency of a multi-level IoT structure including cloud servers, gateways, and sensors. Both toolkits employ low accuracy estimations such as look-up tables for latency and power analysis. *RelIoT* does a finer-grained estimation for different types of IoT devices running various applications.

Reliability Modeling and Management.

Prior work has studied reliability degradation phenomena on processor-based systems. The considered failure mechanisms include Time-Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI) and Electromigration (EM), which all limit device lifetime [33, 14, 12]. In these works, the reliability degradation problem is approached in two steps: (i) 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. (ii) Based on the reliability degradation models, a management algorithm is designed to optimize performance while satisfying reliability constraints. The trade-off between performance and reliability could be adjusted during runtime by voltage scaling [33, 14, 12], task scheduling [82], or both [10]. The recent work by Mercati *et al.* [10] implements the above-mentioned models on a mobile phone, showing as much as a one-year improvement on lifetime with dynamic reliability management. Despite the impressive results on individual devices, reliability management for IoT networks is yet to be investigated. Recently, a dynamic optimization approach was proposed

to manage battery reliability degradation in IoT networks, but their work does not consider the reliability of other device components (e.g., processor) [4].

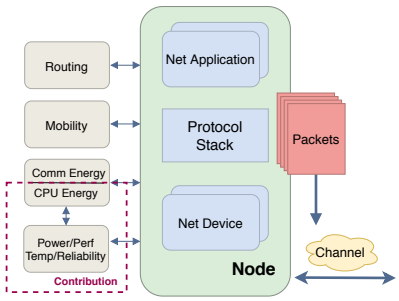
In this thesis, we propose *RelIoT*, a framework for end-to-end reliability simulation in IoT networks to enable investigation of reliability trade-offs and prototyping of reliability management strategies. We develop and integrate power, performance, temperature, and reliability modules into ns-3. In contrast to prior work on network simulators, *RelIoT* offers temperature and reliability estimation for the networked devices.

2.3 Reliability Framework for RelIoT

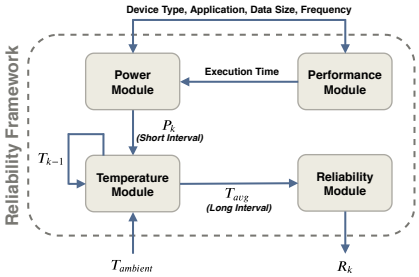
In this section, we give a background on ns-3 and its features, then describe the overall structure of *RelIoT* and its integration with ns-3.

2.3.1 ns-3 Preliminary

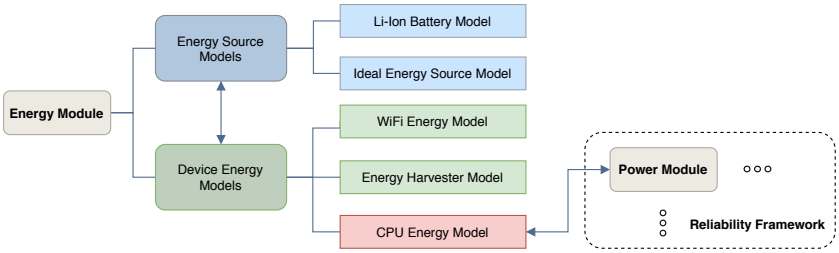
ns-3 is built as a system of libraries that work together to simulate a computer network. To do simulation using ns–3, the user writes a *C++* program that links the various elements from the library needed to describe the communication network being simulated. ns–3 has a library of *objects* for all of the various elements that comprise a network (*objects* are highlighted in italics). *Nodes* are a representation of computing devices that connect to a network. Sensors, routers, hubs, gateways, and servers in the IoT architecture can be all considered *Node* objects. Fig. 2.1a shows the structure of a typical ns-3 *Node*. *Net Devices* represent the physical device that connects a *Node* to a communications *Channel*. For example, the *Net Device* can be a simple Ethernet network interface card or a wireless IEEE 802.11 device, and the *Channel* could be a fiber–optic link or the wireless spectrum. *Packets* are the fundamental unit of information exchange in a network. A *Packet* contains headers describing the information needed by the protocol implementation and a payload which represents the actual data being communicated between network devices. Each protocol in the *Protocol Stack* performs some operation on network packets and then passes them to another layer in the stack for additional processing. The



(a) ns-3 node structure



(b) Proposed reliability framework



(c) Energy module and power module interconnection

Figure 2.1. ns-3 integration of the reliability framework

Net Applications are simple networking applications that specify the attributes of communication policies between devices. All of these individual components are aggregated on the *Node* objects to give them communication ability and set up networking activity. Other modules, such as *Routing*, *Mobility*, and *Energy*, can be installed to provide additional functionality to *Nodes*.

2.3.2 Overview of the Proposed Framework

Our proposed framework consists of separate modules for power, performance, temperature, and reliability, as shown in Fig. 2.1b. IoT devices can run some applications to process the sensed or collected data before sending it to a central entity. In this case, as soon as an application starts, the performance module first calculates its execution time. Then, the power module gives an estimate of power consumption within the execution interval of the application. If the IoT device is not running any applications, then idle power consumption is estimated. Given the power estimation, ambient temperature, the temperature module outputs an estimated temperature, which is fed to the reliability module. Finally, reliability is calculated based on temperature. The modules operate on two different time scales: *Long Intervals*, on the order

of days that it takes for reliability to change, and *Short Intervals*, on the order of milliseconds. Both performance and power values are updated every *Short Interval*. Reliability estimation is computationally expensive, so it is only done once every *Long Interval* using the average temperature over each interval. The underlying mechanisms of each module are be explained in Section 2.4.

2.3.3 Integration with ns-3

As shown in Fig. 2.1a, our framework is implemented as an additional set of modules that can be aggregated on the *Nodes*, adhering to the structure and conventions of ns-3. The power and performance modules provide functions to other modules for querying power consumption and execution time values. We also provide an interface connecting the ns-3 energy module and our reliability framework (Fig. 2.1c). The energy module (proposed in [20]) consists of a set of energy sources and device energy models. An *EnergySourceModel* is an abstraction for the power supply (e.g. battery) of a *Node*. The *DeviceEnergyModel*s represent energy consuming components of a *Node*, for example, a WiFi radio. We implement a model called *CPUEnergyModel* as a child class of *DeviceEnergyModel*. The features of *CPUEnergyModel* are as follows:

- It is designed to be state-based, where the CPU can take the states *Idle* or *Busy*. The CPU will be *Busy* while processing packets received, e.g., while executing some applications such as encryption, decryption, compression, or Machine Learning (ML) algorithms.
- To determine when a transition occurs between states, a *PhyListener* is used. In ns-3, *PhyListener* is an object that monitors the network packet transmissions and receptions at the physical (PHY) layer. After an *Idle* node completes receiving data of specified size, the *PhyListener* notifies *CPUEnergyModel*. Then, the specified application is executed and state is set to *Busy*.
- It calculates the total energy consumed according to the power consumption value acquired

from the power module and the execution time value acquired from the performance module. Then, it updates the remaining energy of the energy source described by *EnergySourceModel*.

2.4 Modules and Device Behavior Modeling

In this section, we describe the functionality of the proposed modules and interfaces and present underlying models in detail.

2.4.1 Power Module

The power module supports functions for running and terminating an application and switching between CPU states *Idle* and *Busy*. The value of power consumption is updated at a predefined period *Short Interval*, according to the selected power model. The power and temperature modules are interconnected; power consumption updates subsequently lead to temperature updates.

From cycle-accurate, instruction-level analysis to functional-level analysis, there are numerous power modeling techniques at different levels of abstraction. Low-level models use a fine-grain representation of the CPU, which usually implies that the time required for power estimation is large due to high computational complexity. This is undesirable for network simulations because it becomes very time consuming to simulate networks with a great number of nodes. In our framework, we offer two CPU power models having low model complexity while still providing good estimation accuracy. To improve the extensibility of the simulator for custom applications, we have included functionality for users to add new models to the power module. Parameters of the power models are configurable through external interfaces.

Frequency & Utilization-based Power Model. The idea of estimating CPU power consumption on embedded devices based on CPU frequency and utilization is well studied. In a previous work [83], the authors use a linear combination of frequency and utilization to characterize the CPU power of a smartphone, achieving less than 2.5% average error. Similarly, we use

linear models in our simulator to predict CPU power consumption P_{CPU}. The equation is given as:

$$P_{CPU}(t) = a \cdot f(t) + b \cdot u(t) + c \tag{2.1}$$

where $f(t)$ and $u(t)$ are CPU frequency and utilization at time t respectively. The coefficients a, b, c are learned through linear regression based on datasets collected on real devices. The frequency & utilization-based power model provides a good estimation accuracy for CPU power estimation on embedded devices. However, it requires frequency and utilization traces as inputs to the simulator which might not be available in practice.

Application-based Power Model. The power consumption of embedded devices varies depending on the running application. An application-based model is convenient when there are only high-level functional properties are available, e.g. input data size. Different applications have different power trends (i.e. linear, exponential, etc.) as the size of input data increases. Furthermore, the power consumed by running the same applications varies for different devices. In our framework, we adopt the modeling methodology proposed in [41], where the authors characterize and verify power models of running ML algorithms on edge devices (i.e. Raspberry Pi) and servers. They train, test, and cross-validate four regression models (linear, polynomial, log, and exponential regression), and select the best one. The input is the size of processed data by the application and the output is power consumption for these models. We leverage this methodology to deploy models for Raspberry Pi's and servers, but also apply the same methodology to build our own models for microcontrollers such as Arduinos. In addition to the 22 ML algorithms modeled in [41], our framework delivers a CPU power model for Multilayer Perceptron (MLP) based on the number of MAC (multiply-accumulate) operations. The same modeling approach can be applied to other neural network architectures such as Convolutional Neural Networks (CNNs).

2.4.2 Performance Module

IoT systems usually need to satisfy some performance requirements to provide adequate Quality of Service (QoS). To evaluate and monitor the performance of deployed applications and hence the overall network, we implement a performance module. Various metrics can be used to quantify performance, e.g., throughput, response time, etc. The performance metric is application-specific. For example, delay and throughput are critical in multimedia streaming applications whereas information accuracy is the main criterion for performance in some ML applications.

In our current release, we provide an *Execution Time Model*. We use the input data size of the application or number of MAC operations it needs to perform to estimate the application execution time. To build the model, we measure the execution times of various applications on a target device, then fit regression models to the collected data. Certain performance metrics can be calculated using the execution time value. For example, let t_{exec} be the execution time of an application, then its throughput can be obtained as D/t_{exec} where D is the input data size. In addition, end-to-end delay of a network path can be computed as the sum of communication and computation delays among the path (communication delay can be obtained using default ns-3 modules).

For both the power and performance modules, users are able to configure coefficients of the existing model or add new models with provided APIs.

2.4.3 Temperature Module

The goal of the temperature module is to estimate CPU temperature (based on CPU power consumption and ambient temperature) and to calculate the average temperature over a *Long Interval*. We adopt a thermal modeling strategy that can be used for any IoT device.

We assume that we do not have knowledge about the information describing topological and physical parameters of the device (e.g., we do not know material characteristics and layers

of the devices' PCB board) so we cannot do a physical simulation of the process. To have an acceptable level of complexity in our simulator, we work on high-level information gathered from the coarse-grained thermal sensors of the device's key heat sources. Such information is available in most of the devices today like smartphones and single-board computers (e.g., Raspberry Pi).

Let the number of the heat sources be n and let $T_k \in \mathbb{R}^n$ represent the vector of temperatures observed by thermal sensors and $P_k \in \mathbb{R}^n$ be the power consumed by the heat sources at time instant k. Each heat source is assumed to have one thermal sensor measuring its temperature. Then, temperature T_{k+1} at time instant $k+1$ can be predicted given the current temperature T_k and power P_k at time k. The discrete-time state-space model of the device's thermal behavior is expressed in Equation (2.2) [84].

$$T_{k+1} = A \cdot T_k + B \cdot P_k + C \cdot T_k^{env} \tag{2.2}$$

where $A, B \in \mathbb{R}^{n \times n}$ are defined as the state and the input matrices. T_k^{env} is the ambient temperature and C is a vector of coefficients which weighs the impact of ambient temperature on each heat source's internal temperature. We use system identification methods to derive the model from measured power and temperature traces. A, B and C parameters are different for each class of devices, so we offer multiple device thermal models and made the parameters configurable through the temperature module API. The order of the model is equal to the number of the heat sources n. In our initial work, we use $n = 1$, where the only source is CPU. However, the extension to multiple sources is straightforward in our framework. For example, if a power model for GPU is provided, then power consumption values from both CPU and GPU can be used to predict temperature.

The temperature module updates the states in Equation (2.2) at a time resolution of *Short Interval*, the same time granularity as power estimation updates. On the other hand, average temperature $\bar{T}$ is calculated for every *Long Interval* denoted *LI*. $\bar{T}$ is the exponential moving

average of past temperature values in the interval k to $k+LI$.

$$\bar{T}_{k+1} = \alpha \cdot T_k - (1-\alpha) \cdot \bar{T}_{LI} \tag{2.3}$$

where α is a weighing coefficient that is configured depending on the length of interval LI.

2.4.4 Reliability Module

The reliability module is the last component in the power, temperature, reliability module hierarchy. Temperature is estimated using power, while reliability is estimated using temperature. Unlike power and temperature, reliability is a slowly changing variable. Therefore, reliability can be estimated on a longer time scale, on the order of hours or days. Reliability degradation is affected more by average stress over a long time interval rather than instantaneous stress. We leverage these properties to calculate reliability sparsely because reliability models are highly compute-intensive. The reliability module does estimation every *Long Interval*, using temperatures averaged over the interval. It polls the temperature module to fetch the average temperature $\bar{T}_{LI}$ every LI, then $\bar{T}_{LI}$ is reset to start a new averaging operation.

Reliability is defined as the probability of not having failures before a given time t. To obtain the overall reliability of a processor, the effects of different failure mechanisms should be combined. We use the sum-of-failure-rates model as in RAMP [33], which states that the processor is a series failure system; the first instance of a failure due to any mechanism causes the entire processor to fail. In our reliability model, the single device reliability is a product of the reliabilities due to different failure mechanisms such as Time Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI), Electromigration (EM) and Thermal Cycling (TC). These mechanisms all depend on thermals.

Time Dependent Dielectric Breakdown (TDDB) Reliability Model. The thin gate oxide layer in transistors introduces a risk of breakdown and shortening devices lifetime. Due to gate oxide degradation, which is a non-reversible mechanism with a cumulatively increasing impact,

a breakdown occurs. The reliability of a single transistor i with oxide thickness x_i subject to oxide degradation can be expressed as [85]:

$$R_i(t) = e^{-a(\frac{t}{\gamma})^{\beta x_i}} \tag{2.4}$$

where t is the time-to-breakdown, a is the device area normalized with respect to the minimum area, and γ and β are respectively the scale parameter and shape parameter. The scale parameter γ represents the characteristic life, which is the time where 63.2% of devices fail, and it depends on voltage and temperature. The shape parameter β, instead, is a function of the critical defect density, which in turn depends on oxide thickness, temperature and applied voltage. $R(t)$ is a monotonically decreasing function with values in the range of [0, 1] indicating the probability that the system will not fail.

The reliability of the entire chip R_C can be expressed as the product of single transistor reliabilities:

$$R_C(t) = \prod_{i=1}^{m} R_i(t) = e^{\sum_{i=1}^{m} -a_i(\frac{t}{\gamma_i})^{\beta_i x_i}} \tag{2.5}$$

m is the number of transistors on the chip which can be on the order of millions. Since different regions of the chip have similar temperatures, the complexity possessed by large m on the computation of Equation (2.5) can be reduced by assuming the same scale and shape parameters over the chip[12].

The R_C expression in Equation (2.5) assumes a constant temperature applied from time $t = 0$, thus it is only representative of static systems. To capture the dynamics of reliability under varying temperature, we discretize the time and calculate reliability at each time step as shown in Equation (2.6). The temperature is assumed to be constant between discrete time steps.

$$R_k = R_{k-1} - \Big(R_C(t_{k-1}, T_{k-1,k}) - R_C(t_k, T_{k-1,k})\Big) \tag{2.6}$$

In Equation (2.6), k indicates the k^{th} time instant and $T_{k-1,k}$ is the temperature experienced by

the chip between the time instants $k-1$ and k. We set this interval between adjacent time steps as the *Long Interval* and let $T_{k-1,k}$ be equal to the average temperature $\bar{T}_{LI}$ of the corresponding *LI*.

The reliability module can work with any failure mechanism or combination of multiple mechanisms as long as the mechanism can be described by a function $R_C(t)$, as in Equation (2.5). For example, the module can be extended to include NBTI and HCI if we describe the reliability functions associated with these mechanisms, respectively R_{NBTI} and R_{HCI}. Then, by the sum-of-failure-rates approach, the reliability module calculates the total system reliability as the product of the functions associated with the single mechanisms as $R_C(t) = R_{TDDB}(t) \cdot R_{NBTI}(t) \cdot R_{HCI}(t)$. Equation (2.6) would not need any modifications since it is general and does not depend on a specific $R_C(t)$.

2.5 Experiments and Results

In this section, we first present validation results on a three-node network topology, comparing power, performance, and temperature measurements from experiments with the simulated traces. We then use a testbed with a mesh network of 10 heterogeneous nodes to evaluate the accuracy of the simulator under different networking conditions and temperatures. We cannot explicitly validate reliability because it requires long term experiments and specialized degradation sensors. Finally, we illustrate how the proposed simulator is useful in exploring energy, performance, reliability trade-offs in a network and show that it can be used to implement reliability-aware strategies. We analyze examples of energy-optimized and reliability-optimized network configurations to motivate reliability-aware network design and management.

2.5.1 Validation and Evaluation

Three-Node Network Topology.

To validate the device models and to verify the functionality of the simulator modules, we use a simple three-node network. The setup consists of an ESP8266 WiFi microchip with

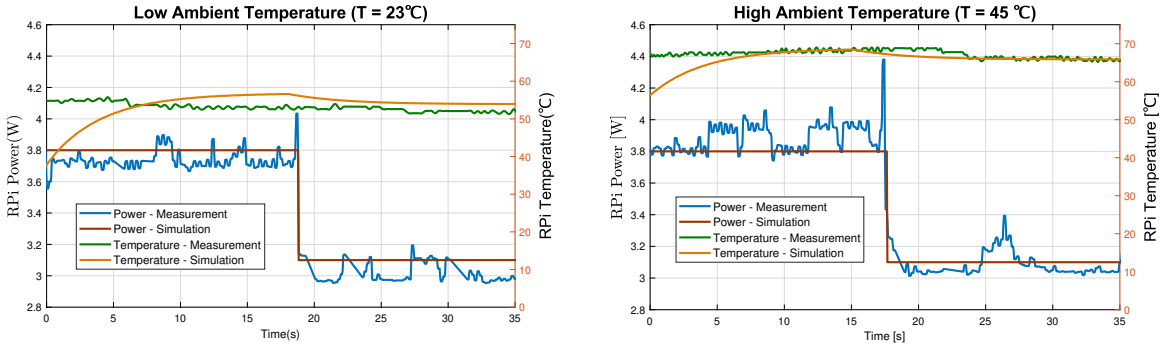


Figure 2.2. RPi3 power and temperature traces

microcontroller, a Raspberry Pi 3 (RPi3), and a PC. The devices communicate over WiFi (IEEE 802.11b) and transmit/receive TCP/IP packets using MQTT protocol [86]. The ESP8266 samples random data as a sensor node, runs median filtering to preprocess the data, and sends filtered data to RPi3. The data is further processed by an application on the RPi3, or the computation can be offloaded to the PC. This type of computation offloading is common in IoT edge devices and is representative of their usual operation [87]. If the application is chosen to be offloaded, then the RPi3 is only responsible of relaying incoming data to the PC.

In our three-node experiments, we collected 5 different measurements synchronously:

(i) RPi3 power consumption (via HIOKI 3334 power meter [88]),

(ii) RPi3 CPU temperature (via built-in temperature sensor),

(iii) ESP8266 power consumption (via INA219 power monitor [89]),

(iv) ESP8266 CPU temperature (via built-in temperature sensor),

(v) Ambient temperature (via DHT22 temperature sensor [90]).

Measurements and simulation results are presented in Fig. 2.2 for an example test case under two different ambient temperatures. The goal here is to show a temporal view of the simulator output, particularly in a dynamic case where the simulated device has a varying workload. In this experiment, the RPi3 runs a data processing application with incoming data

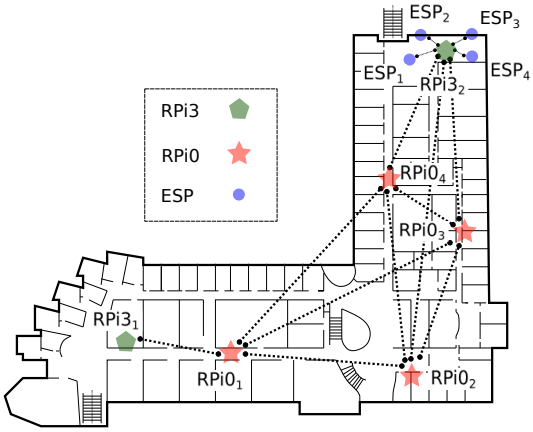


Figure 2.3. Mesh network topology

input from ESP8266 for the first 15-20 seconds. After that, the application is offloaded to the PC and the RPi3 only relays data while its CPU is *idle*. As shown in Fig. 2.2, the simulator output follows the real power and temperature traces with a mean error of 3.42% and 6.19% in low ambient temperature, and with a mean error of 2.69% and 3.97% in high ambient temperature. The discrepancy between real and simulated temperatures at the beginning of each plot is because of the initial condition set for the temperature in the simulator. The temperature starts from a lower initial condition and reaches a steady-state value.

Overall, applying the same modeling methodology of reference [41], we estimate the execution time and energy consumption of the RPi3 for 23 different ML applications with average errors of 3.8% and 4.5%, respectively. For the CPU temperature, the state-space model predictions stays within $\pm 1.5°C$ of measurements at steady-state, for all applications.

Mesh Network Topology.

To show that our simulator can correctly capture devices' behavior in a more complicated scenario, we simulate a larger network under different configurations and operating conditions, then validate it using our testbed. As shown in Fig. 2.3, the testbed spans a whole floor in UCSD CSE department building, including two Raspberry Pi 3 (RPi3), four Raspberry Pi 0 (RPi0) and four ESP8266. Data is generated from each node and communicated to the sink node

$RPi3_1$ in multiple hops via MQTT. The network of all RPis works in an ad-hoc manner, while all ESP8266s forward their data to $RPi3_2$ that is a gateway for that local area. Not all devices can communicate with each other because some pairs are out of communication range. The connections are depicted in Fig. 2.3. Using this setup, we both implement and simulate following scenarios:

Scenario 1 $RPi3_1$ and $RPi0_1$ process the data, while the other devices only communicate.The ambient temperature for each network device is approximately 25°C.

Scenario 2 The same devices process data as Scenario 1. We use a heater that raises the ambient temperature around $RPi0_1$ to 37 °C, while the rest of the devices are in the normal ambient condition of 25 °C.

Scenario 3 The data processing duties of $RPi3_1$ and $RPi0_1$ are distributed between $RPi0_2$, $RPi0_3$, and $RPi0_4$. Therefore, each of these three devices only transmits the outputs of data processing tasks to $RPi0_1$, which directly forwards them to $RPi3_1$. $RPi0_1$ is still in a heated environment of 37 °C.

An ML application can be split and distributed to edge devices, which allows us to realize the different configurations in these scenarios for allocating data processing without changing the overall application behavior [91, 92]. Fig. 2.4 illustrate the power and temperature distribution of the devices, while Fig. 2.5 shows the simulated reliability traces in a year. We only depict the measurement and simulation statistics on *RPi0s* in each scenario, but the rest show similar trends. Comparing collected traces to simulation logs, our result shows that *RelIoT* is able to estimate average power within ± 0.11 W ($\sim$11%), and average temperature within ± 4 °*C* ($\sim$9%). It can be seen from Fig. 2.4 that, although extremities in both power and temperature are difficult to track, *RelIoT* is able to precisely capture the averages in different configurations. Scenario 3 distributes the workload to other *RPi0s*, thus significantly reduces the network traffic. Consequently, power and temperature of both $RPi0_1$ and the rest *RPi0s* drop, which is also reflected in the simulation

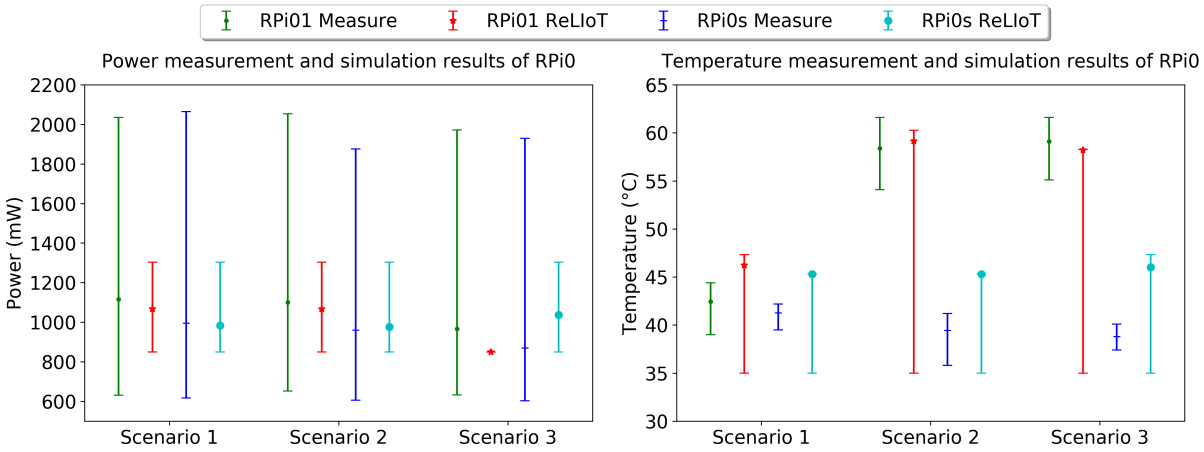


Figure 2.4. Collected and simulated statistics of RPi0s (average, max, and min values).

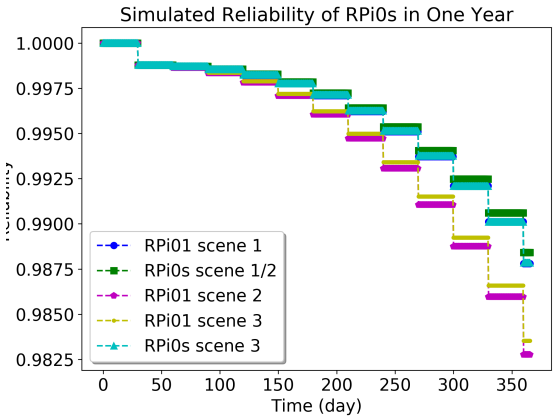


Figure 2.5. Reliability degradation of RPi0s in one year.

RelIoT starts simulation from a device temperature of 35 °C, which explains why the minimum temperature of *RelIoT* consistently locates at 35 °C.

The power and temperature validation experiments lasts 300s, but we simulate the network for a time-span of one year to observe the long-term reliability changes. The stair pattern in Fig. 2.5 is a result of *RelIoT* updating the reliability by the end of each *Long Interval*. In all scenarios, reliability remains fairly high if the device is in normal ambient temperature. In Scenario 1, $RPi0_1$ degrades slightly faster than the rest of the $RPi0_s$ due to its data processing workload. However, in Scenario 2, the raised ambient temperature together with its workload lead to a drastic degradation in reliability. In such case, workload reallocation as in Scenario

3 can alleviate degradation. The network devices in environments with low temperatures can take on a higher workload to mitigate reliability problems of the quickly degrading devices. The result in Fig. 2.5 implies the necessity for reliability-aware management in IoT networks.

2.5.2 Reliability-Aware Management

Most of the IoT devices are battery-powered and/or rely on energy harvesters with limited energy sources. Therefore, traditionally, many network management solutions aim at optimizing the energy consumption while satisfying some Quality-of-Service (QoS) constraints (throughput, delay, jitter, network coverage, etc.). In this context, reliability is also a design parameter that can be optimized or a certain overall reliability constraint can be subjected to the network. Although correlated, the optimal energy efficiency and reliability usually are not ensured by the same management strategy. The designers need to find good trade-offs between energy savings and reliability. In this section, we show how our simulator addresses this issue by making reliability-aware management and design possible. To emphasize the differences between two approaches and to motivate reliability-aware management, we provide simulation results for different scenarios of energy-optimized and reliability-optimized network management strategies using the topology in Fig. 2.3.

Energy-Optimized.

Our interest here is to partition an application into smaller tasks and find the task allocation that maximizes the lifetime of a network. Many ML applications can be partitioned while preserving functionality [91, 92]. For each device in the network, the energy consumed for computing and communicating data of size s is given as:

$$\text{Computation:} \qquad P^{\xi}_{device}(s) \times t_{exec}(s) \tag{2.7}$$

$$\text{Communication:} \qquad P_{wifi}(d, BW) \times \frac{s}{BW} \tag{2.8}$$

where ξ denotes the application, t_{exec} is the application execution time, d denotes the communication distance, and BW is the communication bandwidth. $P_{wifi}(d, BW)$ is the power consumption of WiFi which can be parameterized by distance and bandwidth allocation BW [41]. In an energy-optimized application partition, the mapping of tasks to the devices depends on:

(i) Power characteristics of the application,

(ii) Execution time,

(iii) Allocated bandwidth,

(iv) Distance between the neighbouring devices.

We adopt the convex optimization formulation from [93] and apply it to our problem, with a slight modification by adding the computation energy term in Equation (2.7). We find the optimal partitioning of the application such that the *maximum* energy consumption among network devices due to computation and communication of the data is minimized.

Reliability-Optimized.

Similar to the previous case, we map the tasks of an application to the network devices. We use the same solution approach, but this time, the objective is to maximize the *minimum* reliability among network devices. Reliability of each device is $R_{C,device}(t, T)$, which is dependent on time and temperature. We simulate a time horizon t_{sim}, so we want to optimize for $R_{C,device}(t_{sim}, T)$. This is under the assumption of environment temperature T_{amb} being constant for the entire horizon. In the following experiments, a static solution (constant for the whole time horizon) is simulated for both energy-optimized and reliability-optimized cases, but it can be made dynamic by solving for the current energy and reliability estimates at each time instant, as in (2.6). In this way, the solution can adapt to changing network configurations (bandwidth, applications) and operating conditions (environment temperature).

Fig. 2.6 presents the comparison of energy-optimized and reliability-optimized solutions for different bandwidth and environment temperature configurations. The network devices

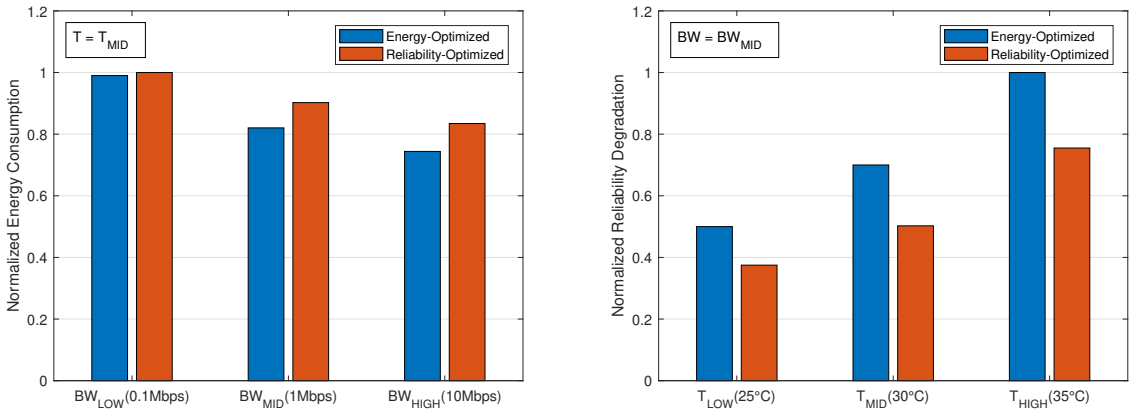


Figure 2.6. Energy consumption and reliability degradation results for two approaches

run a part of a data processing application where the optimal partitions are determined by the two approaches. The energy-optimized partition brings 1.0%, 9.1%, and 10.9% better energy efficiency compared to the reliability-optimized partition for $0.1Mbps$, $1Mbps$, and $10Mbps$ bandwidth configurations respectively. Referring to Equation (2.8), it can be seen that as bandwidth increases, the time it takes to communicate data of size s decreases, hence, decreasing the communication energy. The difference of energy efficiencies between two solutions are increasing with bandwidth because the energy-optimized solution leverages the decrease in communication energy and allocates more communication instead of computation to the higher energy consuming network devices. On the other hand, the reliability-optimized partition results in 25.0% , 28.2%, and 24.5% less reliability degradation compared to the energy-optimized partition for $25°C$, $30°C$, and $35°C$ environment temperatures respectively. The reliability-optimized solution allocates less computation on the most degrading network devices, conserving reliability. These results show that, although being correlated, the optimal energy efficiency and reliability do not yield from the same management strategy. Therefore, if the concern is particularly the reliability, a reliability-aware management strategy should be adopted.

2.6 Conclusion

We presented a novel framework for the reliability analysis of IoT networks using the ns-3 network simulator. The proposed framework can be used to explore trade-offs between power, performance, and reliability of devices in a network. We validated our reliability framework in two experimental setups: a three-node network and a ten-node mesh network. Additionally, we motivated the need for reliability-aware management through example simulation results of energy-optimized and reliability-optimized management strategies. As future work, we plan to leverage our framework for design space exploration (DSE) of IoT networks. We can simulate, explore, and check the feasibility of different network configurations in terms of different objectives such as energy efficiency, reliability, and performance. We believe that our contribution will help researchers to study the reliability degradation problem in large-scale networks.

Chapter 2 contains material from "*RelIoT*: Reliability Simulator for IoT Networks", by Kazim Ergun, Xiaofan Yu, Nitish Nagesh, Lucy Cherkasova, Pietro Mercati, Raid Ayoub, Tajana Rosing, which appears in International Conference on Internet of Things (ICIOT), 2020 [1]. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Dynamic Reliability Management of IoT Edge Computing Systems

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 chapter, 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.

3.1 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 [6]. A significant portion of spending on the IoT ($746 billion in 2019 [8]) is associated with maintenance and technical diagnostics due to system failures, which motivates our work.

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 [7]. 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) [11, 55, 12]. 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. 3.1a we depict the steady-state temperature of a device as a function of its power dissipation at various ambient temperatures. Also, Fig. 3.1b shows the reliability over time of

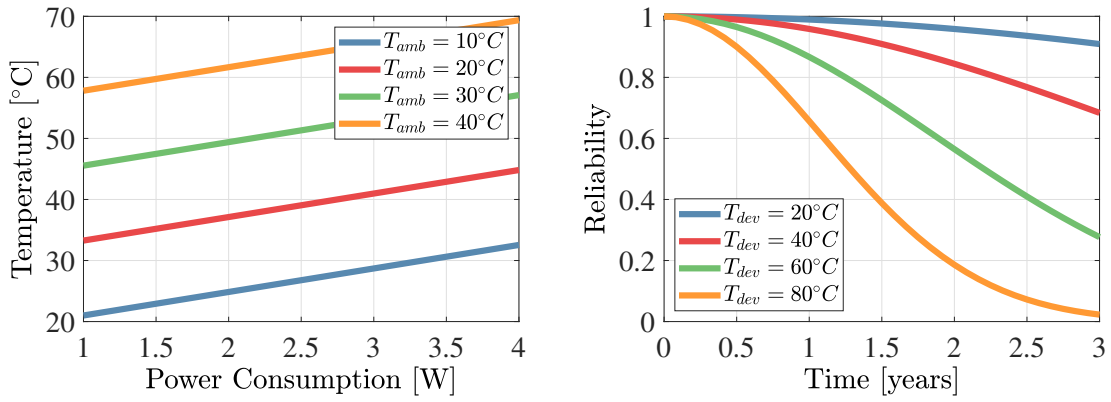


Figure 3.1. (a) Device temperature as a function of power dissipation at different ambient temperatures (b) Device reliability over time

the same device as a function of its temperature. The values are based on our measurements in Section 3.8. (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) [94, 95], Edge Computing [25] is emerging as a promising solution, where data processing is pushed to the edge of the IoT network (as shown in Fig. 3.2). 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 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 [33]. Curbing power dissipation, in particular, helps by lowering the device temperatures and reducing the effect of temperature-driven failure mechanisms [55].

The Edge Computing architecture utilizes gateways to enable application-specific con-

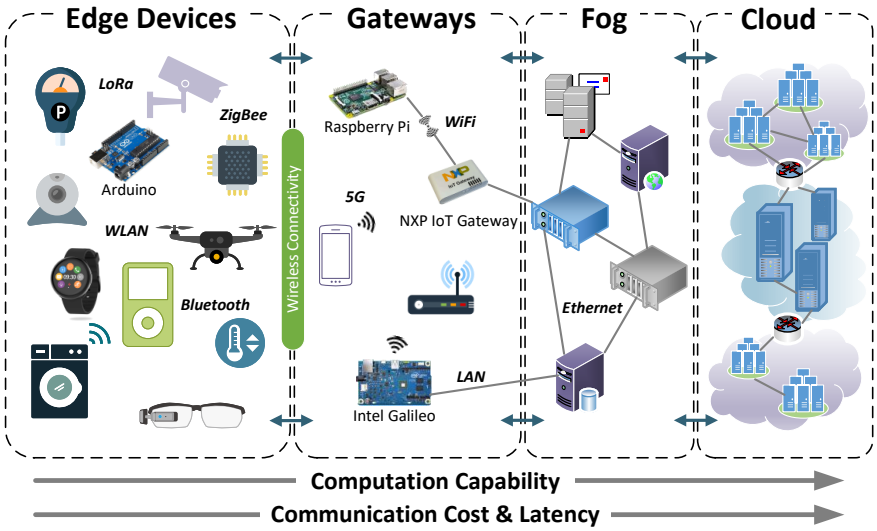


Figure 3.2. IoT network architecture

nectivity between edge and fog devices (Fig. 3.2) [94]. 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. 3.2, 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 [38, 39, 40] 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 [41, 42, 38, 43], 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 [44]. 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 [40], with no or minimum dependency on the cloud [94]. 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 thesis, 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 work 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.

3.2 Related Work

3.2.1 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 [9]. The majority of previous works dealt with traditional QoS attributes such as service delay and throughput. In [96], 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 [97] 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 [41] and [45] 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 [46], 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 [42]. Similar problems for partitioning and offloading workloads to fog/cloud were solved by game-theoretic approaches [98], multi-objective optimization [47], heuristic algorithms [43], and primal decomposition [99]. 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 [48], 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 [40] denominates these distinct levels as 'operation modes' and advances the prior

work in terms of execution time and memory overhead. Finally, task allocation [49] and task scheduling [39] schemes were proposed to determine where and in which order to execute tasks. In contrast to other works, reference [39] 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 [49, 47, 39, 100]. 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 [16] 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 work extends and improves [16] 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 [101] 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.

3.2.2 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) [50], software reliability issues [51], or batteries running out of energy [52, 53]. For example, in [51], 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 [54]), 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 [102], 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 [33, 14, 12]. 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 [55, 33, 14, 12], task scheduling [82], or both [10]. In [103], a task allocation scheme is presented for multi-processor SoCs which

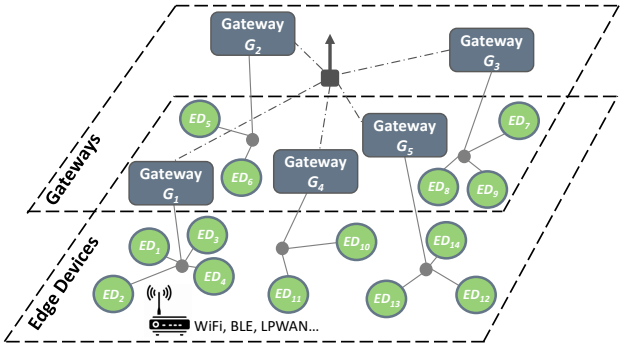


Figure 3.3. Multi-gateway IoT network

maximizes the time to failure of an SoC subject to performance constraints. The authors in [10] implement the above-mentioned mechanisms on a mobile device, showing as much as a one-year improvement on lifetime with dynamic reliability management.

Despite the impressive results on individual devices, reliability management for networks of IoT devices is an open problem. The recent paper in [44] 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 [16], 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.

3.3 System Model

The envisioned IoT network architecture has multiple layers comprising edge devices, gateways, fog, and cloud servers as illustrated in Fig. 3.2. 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.

3.3.1 Network Architecture

We consider an IoT network composed of N edge devices $ED = \{ED_1, ED_2, ..., ED_N\}$ and M gateways $G = \{G_1, G_2, ..., G_M\}$. Each gateway G_j has a subset of $N_j < N$ associated edge devices, which together form a *local network* as shown in Fig. 3.3. 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

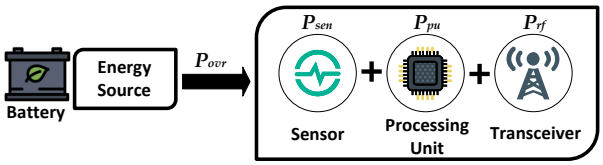


Figure 3.4. IoT device model

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.

3.3.2 Device Models

As depicted in Fig. 3.4, 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.

$$P_{ovr} = P_{sen} + P_{pu} + P_{rf} \tag{3.1}$$

The power consumption P_{pu} of a processing unit can be modeled through Equation (3.2) 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 [104].

$$P_{pu} = P_{dyn} + P_{lea} = \alpha C^{eff} V_{dd}^2 f + V_{dd}(b_T T^2 e^{\frac{k}{T}} + I_{gate}) \tag{3.2}$$

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} [105], 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 [106] and the required power P^{rf} to transmit L bits/second is governed by:

$$P_{rf} = \rho_1(d)\frac{L}{g} + \rho_2 \tag{3.3}$$

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.

$$P_{sen} = c_s \lambda \tag{3.4}$$

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 [107]. 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) [108], 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 [107]. 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 [65] 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 (3.5) 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 [108].

$$\begin{cases} \dfrac{dq_A}{dt} = -\kappa(1-c)q_A + (\kappa c)q_B - \dfrac{P_{ovr}}{V_{dd}} \\ \dfrac{dq_B}{dt} = \kappa(1-c)q_A - (\kappa c)q_B \end{cases} \tag{3.5}$$

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 [84].

$$T_{ed}(k+1) = A_T \cdot T_{ed}(k) + B_T \cdot P_{ed}(k) + C_T \cdot T_{amb}(k) \tag{3.6}$$

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 (3.6) 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 [109]) 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) [33]. 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 (3.7).

$$MTTF_{TDDB} = A_0 \exp - \gamma E_{ox} \exp \frac{E_a}{k_B T} \tag{3.7}$$

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:

$$MTTF_{NBTI} = A_0 (\frac{1}{V})^{\gamma_v} \exp \frac{E_a}{k_B T} \tag{3.8}$$

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 (3.9) for N-channel devices.

$$MTTF_{HCI} = B I_{sub}^{-C_{mat}} \exp \frac{E_a}{k_B T} \tag{3.9}$$

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 [33], 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:

$$MTTF_{ed} = \frac{1}{\sum_{i=1}^{n_s} \sum_{j=1}^{n_m} \frac{1}{MTTF_{ij}}} \tag{3.10}$$

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 (3.11), 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 [10].

$$MTTF = \int_0^\infty R(t)dt \tag{3.11}$$

The reliability function $R(t)$, in general, is expressed as a monotonically decreasing exponential function of time and temperature [55].

$$R(t) = \gamma_1 \exp(-\frac{E_a}{k_B T})\exp(-\gamma_2 t) \tag{3.12}$$

where γ_1, γ_2 are the constants depending on the respective mechanism. The expression in Equation (3.12) is only representative of static systems because it assumes a constant temperature applied from time $t = 0$. The workloads and temperature vary over time, so is the degradation process. Therefore, we introduce *equivalent degradation time* to characterize the reliability degradation effect under such varying conditions. Given the reliability degradation of a device under temperature T_1 for duration t_1, the equivalent degradation under temperature T_2 is described as follows:

$$\Delta R(t_{eqv,1}, T_2) = \Delta R(t_1, T_1) \tag{3.13}$$

The equivalent degradation time $t_{eqv,1}$ can be computed using Equation (3.12). 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.

$$\Delta R(t_{eqv,k-1}, T_{k-1,k}) = R_0 - R_d(k-1) = \Delta R|_{t=t_{k-1}}$$
$$R_d(k) = R(t_{eqv,k-1} + t_{k-1,k}, T_{k-1,k}) \tag{3.14}$$

In Equation (4.3), k indicates the k^{th} time instant and $T_{k-1,k}$ is the temperature experienced by the device between the time instants $k-1$ and k. Similarly, $t_{k-1,k}$ is the time passed between the time instants $k-1$ and k. R_d is the dynamic reliability and R_0 is the reliability of a device at time $t = 0$.

Similar to the system MTTF expression in Equation (3.10), 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.

$$R_{d,ed}(k) = \prod_{i=1}^{n_s} \prod_{j=1}^{n_m} R_{d,ij}(k) \tag{3.15}$$

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.

3.3.3 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 [25]. As illustrated in Fig. 3.5, 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

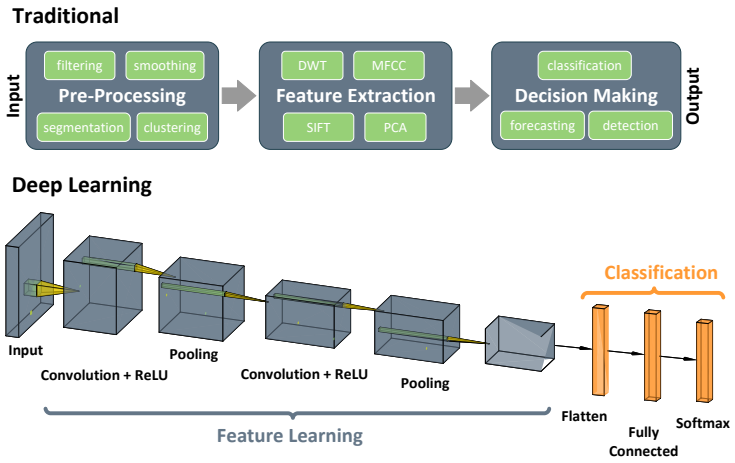


Figure 3.5. Segmented machine learning applications

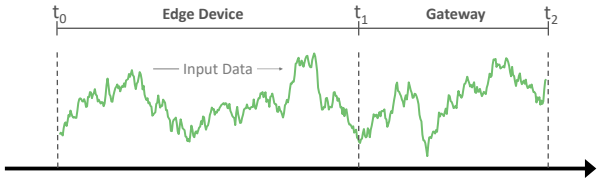


Figure 3.6. Data partitioning and offloading over time

general ML applications [40, 110, 92] and DNNs [111, 112] exist, though, the segmentation of applications and the distribution process are beyond the scope of this thesis. In our work, we assume that this segmentation and distribution process is governed by an external management mechanism, such as [49]. 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. 3.6. 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 [113]) can be breakable into tasks which do not exhibit dependencies across partitions of its input [42]. 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 application, the tasks can be categorized into *delay-sensitive* and *delay-tolerant* (i.e., best-effort) ones [9]. 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.

3.3.4 Network Operation

IoT traffic can be roughly categorized into periodic and event-based modes of communication [114]. 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.

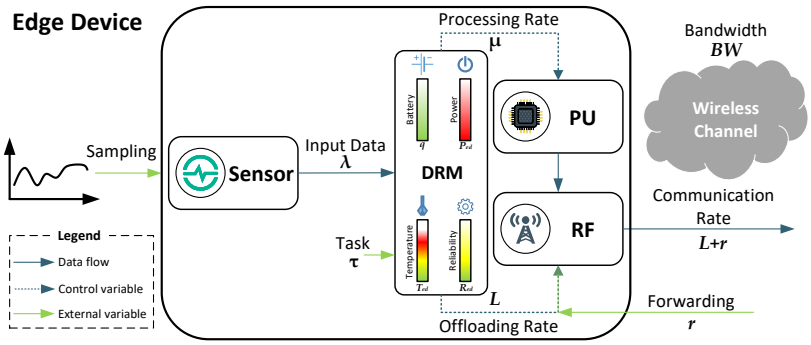


Figure 3.7. Structure of the edge device

The operation of an edge device is illustrated in Fig. 3.7, 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 every edge device in the network. The gateways receive the superposition of offloaded periodic traffics from a number of unsynchronized edge devices (Fig. 3.8). 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 [114]. The computing resources of a gateway

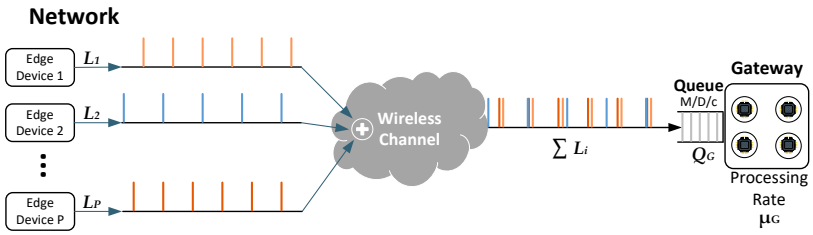


Figure 3.8. Local network operation

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 [115], denoted $Q_{G,j}$. The discrete queue dynamics at the input of the gateways are as follows:

$$Q_{G,j}(k+1) = [Q_{G,j}(k) + \sum L_i(k) - \mu_{G,j}(k)]^+ \tag{3.16}$$

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 (3.16). 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 [116] 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:

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

where $\varepsilon_{G,j}$ 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 $\varepsilon_{G,j}$ 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:

$$W^{max} = [(Q_{G,j}^{max} + Z_{G,j}^{max})/\varepsilon_{G,j}] \tag{3.17}$$

Given the above property, we can choose the appropriate $\varepsilon_{G,j}$ 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.

3.4 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 3.5. Table 5.1 provides the list of symbols that are used in problem formulation, in the order of appearance throughout the chapter.

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 3.5.

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 3.1. 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

3.4.1 Intra-Gateway Problem

The gateways G are only responsible for the edge devices 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:

$$\min_{(\mu,\mathbf{L})} \quad \sum_{k=0}^{T_f-1} -\|\mathbf{1}^T \overline{\mathbf{q}}(k)\|^2 = \sum_{k=0}^{T_f-1} \sum_{i=1}^{N_j} -\|\mathbf{1}^T \mathbf{q}_i(k)\|^2 \tag{3.18}$$

where $(\mu,\mathbf{L}) \triangleq (\mu_1(k),...,\mu_P(k),L_1(k),...,L_P(k))_{k=0}^{T_f-1}$. 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 $R_{d,i}^{sys}$ (Equation 3.15) of each device at the end of the horizon should be at least the target reliability R_{target}, i.e, $R_{d,i}^{sys}(T_f) \geq R^{target}$

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:

$$\min_{\Delta\mathbf{f},\Delta\mathbf{L}} \quad \sum_{k=0}^{T_f-1}\sum_{i=1}^{N_j} -\|\mathbf{1}^T\mathbf{q}_i(k)\|^2 \tag{3.19}$$

$$\text{s.t.} \quad i=1,...,N_j \quad k=0,...,T_f-1$$

$$q_{A,i}(k+1) = A_q q_{A,i}(k) + B_q q_{B,i}(k) - \frac{P_{ovr,i}(k)}{V_{dd}}$$

$$q_{B,i}(k+1) = C_q q_{A,i}(k) + D_q q_{B,i}(k)$$

$$P_{ovr,i}(k) = a_i f_i^3(k) + b_i f_i(k) + c_{s,i}\lambda_i(k) + \rho_1 \frac{L_i(k)+r_i(k)}{g} + \rho_2$$

$$T_{ed,i}(k+1) = A_T T_{ed,i}(k) + B_T P_{ed,i}(k) + C_T T_{amb,i}(k)$$

$$Q_{G,j}(k+1) = [Q_{G,j}(k) + \textstyle\sum_{i=1}^{P} L_i(k) - \mu_{G,j}(k)]^+$$

$$Z_{G,j}(k+1) = [Z_{G,j}(k) - \mu_{G,j}(k) + \varepsilon_{G,j}]^+$$

$$\mu_i(k+1) = \mu_i(k) + d_i\Delta f_i$$

$$L_i(k+1) = L_i(k) + \Delta L_i$$

$$Q_{G,j}(k) < Q_{G,j}^{max}$$

$$Z_{G,j}(k) < Z_{G,j}^{max}$$

$$\textstyle\sum_{i=1}^{P} L_i(k) \le BW_j$$

$$\mu_i(k) + L_i(k) = \lambda_i^{target}(k)$$

$$R_{d,i}(T_f) \ge R^{target}$$

$$|\Delta f_i| \le \Delta f^{max}, \quad |\Delta L_i| \le \Delta L^{max}$$

where we discretized the battery dynamic equations from Equation (3.5) 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.

3.4.2 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 3.3.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.

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 3.7.1. An example solution for our problem

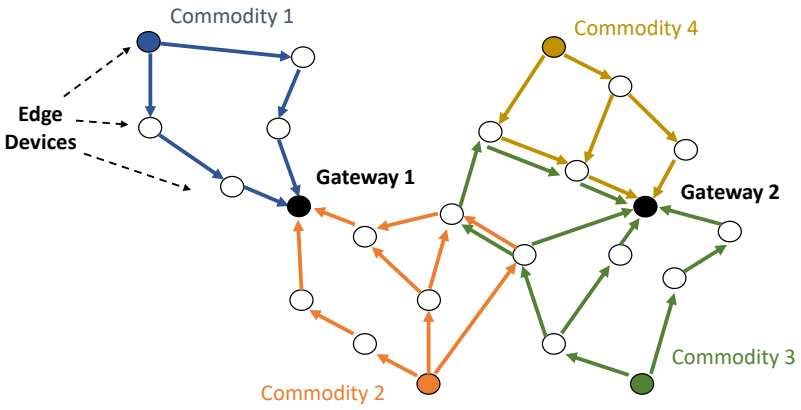


Figure 3.9. Multicommodity flow multiple sink routing

setting is illustrated in Fig. 3.9 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_{kl}^i 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 $\sum_{i=1}^{N} r_{kl}^i$. For simplicity of notation, we stack up all r_{kl} into a single vector and denote network flow as $\mathbf{r} = \{r_{kl}^i\}$, Then, the lifetime of edge device i under flow $\mathbf{r}$ is given by

$$MTTF_{ed,i} = \gamma_c \exp \frac{E_a}{k_B T_{ed,i}(\mathbf{r})} \tag{3.20}$$

We define lifetime in terms of mean time to failure, where Equation (3.20) is a generalized form of MTTF definitions in Equations (3.7), (3.8), (3.9) 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 (3.6)), which in turn relates to data flow through Equation (3.3).

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 [52] and was used in many recent works [117, 118]. In this case, network MTTF under flow $\mathbf{r}$ is the minimum of any node in the network, i.e.

$$MTTF_{net}(\mathbf{r}) = \min_{i \in N} MTTF_{ed,i}(\mathbf{r}) \tag{3.21}$$

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

$$\begin{aligned}
\underset{\mathbf{r},\mathbf{X}_{assign}}{\text{maximize}} \quad & \min_{i \in V_{ED}} MTTF_{ed,i}(\mathbf{r}) \\
\text{subject to} \quad & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = L_i,\ \forall i,k \in V_{ED},\ i = k \\
& \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = 0,\ \forall i,k \in V_{ED},\ i \neq k \\
& r_{kl}^i \geq 0,\ \forall i,k \in V_{ED},\ \forall l \in S_k \\
& \sum_{k \in V_{ED}} r_{kl}^i = L_i,\ \forall i \in V_{ED},\ \forall \{l \in V_G \mid x_{il} = 1\} \\
& \sum_{j=1}^{M} x_{ij} = 1,\ \forall i \in \{1,...,N\} \\
& x_{ij} \in \{0,1\},\ \forall i \in \{1,...,N\}, \forall j \in \{1,...,M\}
\end{aligned} \tag{3.22}$$

The optimization variables are r_{kl}^i 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.

3.5 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. 3.10 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.

- *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

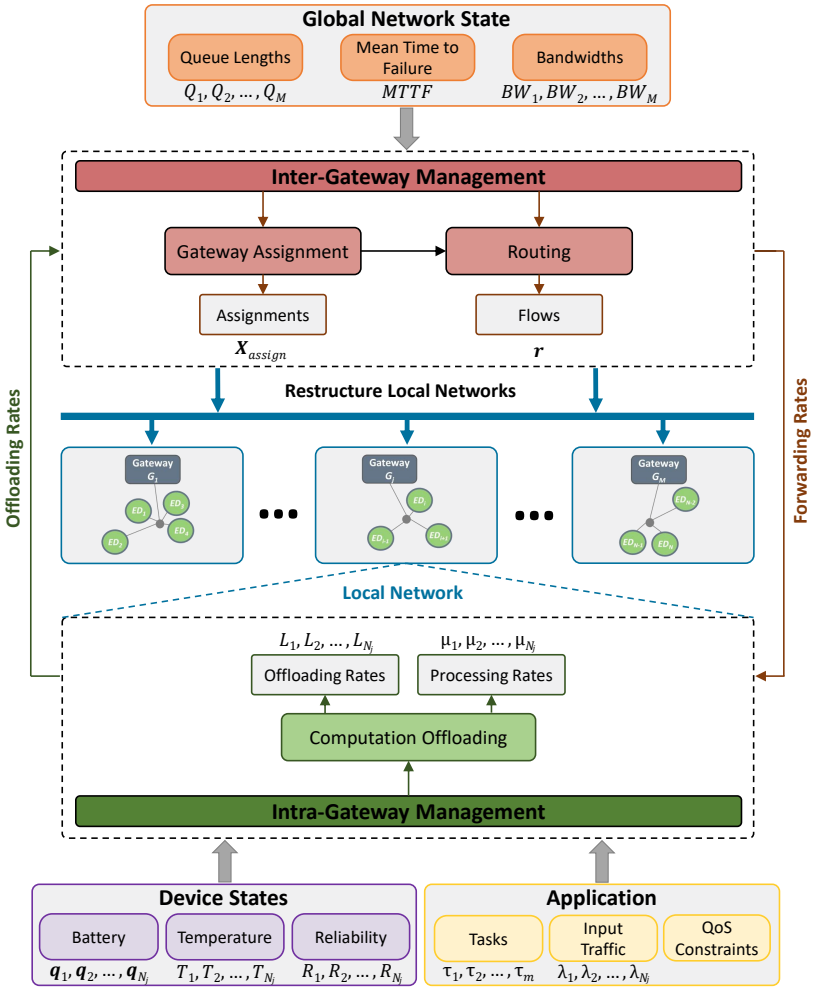


Figure 3.10. Overall architecture of the proposed management scheme

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.

3.6 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.

3.6.1 Centralized MPC

Our problem in Equation (3.19) can be converted to the standard MPC form using the following discrete-time prediction model:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{C}\mathbf{w}(k) \tag{3.23}$$

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.

$$\begin{aligned}
&\min_{\mathbf{u}(k)} \quad \sum_{l=0}^{T_p-1} \sum_{i=1}^{N_j} d(\mathbf{x}(k+l|k), \hat{\mathbf{x}}) \\
&\text{s.t.} \quad i = 1,...,N_j \quad j = 0,...,T_p-1 \\
&\mathbf{x}(k+l+1|k) = \mathbf{A}\mathbf{x}(k+l|k) + \mathbf{B}\mathbf{u}(k+l|k) \\
&\mathbf{D}\mathbf{x}(k+l|k) \leq 0 \\
&\mathbf{E}\mathbf{u}(k+l|k) = 0
\end{aligned} \tag{3.24}$$

The double index notation $(k+l|k)$ in (3.24) 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 $\mathcal{O}(T(n_x^2 + n_x n_u))$ [119], 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.

3.6.2 Proposed Controller Methodology

We decompose the central Nonlinear MPC problem $Pr_C(\mathbf{x}(k))$ into a set of local subproblems $Pr_p(\mathbf{x}_p(k))$, $p \in \{1,...,P\}$ 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 (3.24). 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 $Frequency \rightarrow DissipatedPower \rightarrow Temperature$ can be split into two parts. The first part, as expressed by Equation (3.2), is highly nonlinear while the power-to-temperature model in Equation (3.6) 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 (3.16) and bandwidth constraints. The battery states $q_{A,i}$ and $q_{B,i}$ are coupled through the objective function (3.18) 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.

3.6.3 Proposed Controller Architecture

In the following section, we describe the structure of our proposed controller. Fig. 3.11 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 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 [120]. 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

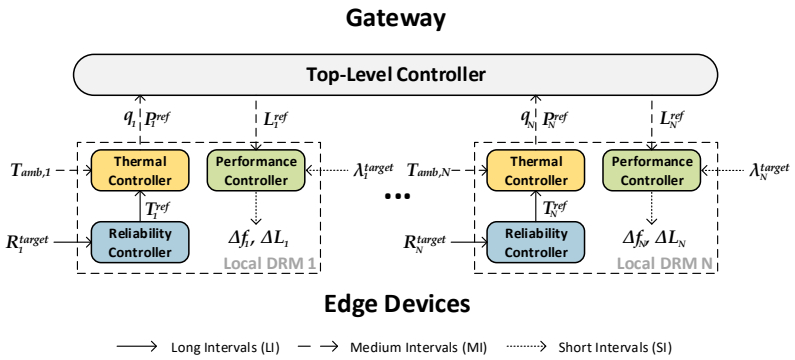


Figure 3.11. Controller block diagram

the target input data rate to obtain a lower, *energy optimal* power reference, which is sent to the *Gateway Top-Level Controller*.

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

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

3.6.4 Edge Reliability Controller

Leveraging the equivalent degradation time technique in Equation (4.3), 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 (3.25). The computation of convex optimization introduces a negligible overhead since the controller activates by intervals in the order of days.

$$\min_{T_{ed}^{target}} \|R(t_{eqv}+t_{rem}, T_{ed}^{ref})\| \tag{3.25}$$

$$\text{s.t.} \quad R(t_{eqv}+t_{rem}, T_{ed}^{target}) \geq R^{target}$$

$$R(t_{eqv}, T_{ed}^{target}) = R_{d,ed}(k_{LI})$$

$R_{d,ed}(k_{LI})$ 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.

$$T_{ed}^{avg}(k_{MI}) = \frac{(k_{MI}-1)\cdot T_{ed}^{avg}(k_{MI}-1) + T_{ed}^{LI}(k_{MI})}{k_{MI}} \tag{3.26}$$

$$T_{ed}^{ref}(k_{MI}) = \frac{t_{LI}\cdot T_{ed}^{target} - k_{MI}\cdot T_{sys}^{avg}(k_{MI})}{t_{LI}-k_{MI}} \tag{3.27}$$

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.

3.6.5 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 (3.6) as follows:

$$T_{ed}(k_{MI}+1) = A_T \cdot T_{ed}(k_{MI}) + B_T \cdot u_T(k_{MI}) \tag{3.28}$$

$$u_T(k_{MI}) = P_{ed}(k_{MI}) + C_T/B_T \cdot T_{amb}(k_{MI}) \tag{3.29}$$

Then, the state feedback $u_T(k_{MI})$ is calculated as We apply the state-feedback control law [121] for a linear system. Then, the input $u_T(k_{MI})$ is calculated as:

$$u_T(k_{MI}) = K_T(T_{ed}^{ref} - T_{ed}(k_{MI})) \tag{3.30}$$

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.

$$P_{ed}(k_{MI}) = K_T(T_{ed}^{ref} - T_{ed}(k_{MI})) - C_T/B_T \cdot T_{amb}(k_{MI}) \tag{3.31}$$

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

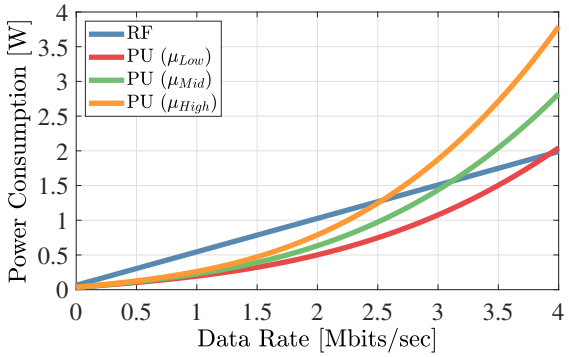


Figure 3.12. Average power as functions of computation and communication rates

Algorithm 1: Power Reference Trimming

Input: λ^{target},$P_{pu}(k_{MI})$,$P_{rf}(k_{MI})$

Output: P_{ed}^{ref}

Calculate P_{pu} for $\mu = \lambda^{target}$

Calculate P_{rf} for $L = \lambda^{target}$

if $P_{pu}|_{\mu=\lambda} < P_{rf}|_{L=\lambda}$ **then**

  if $P_{pu}|_{\mu=\lambda} < P_{pu}(k_{MI})$ **then**

    $P_{ed}^{ref} = [P_{pu}|_{\mu=\lambda}, 0]^T$

  else

    Calculate $\{\mu^{ref} \mid P_{pu}|_{\mu=\mu^{ref}} = P_{pu}(k_{MI})\}$

    $L^{ref} = \lambda^{target} - \mu^{ref}$

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

else

  if $P_{rf}|_{\mu=\lambda} < P_{rf}(k_{MI})$ **then**

    $P_{ed}^{ref} = [0, P_{rf}|_{\mu=\lambda}]^T$

  else

    Calculate $\{L^{ref} \mid P_{rf}|_{L=L^{ref}} = P_{rf}(k_{MI})\}$

    $\mu^{ref} = \lambda^{target} - L^{ref}$

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

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. 3.12. 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 (3.31) 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 (3.31). 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).

3.6.6 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.

$$\min_{\mathbf{P}_{ed}} \quad \sum_{k_{MI}=0}^{T_p-1} -\|\mathbf{1}^T\mathbf{q}(k_{MI})\|_Y^2 + \|\mathbf{P}_{ed}(k_{MI}) - \mathbf{P}_{ed}^{ref}\|_Z^2 \tag{3.32}$$

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

$$q_{A,i}(k_{MI}+1) = A_q q_{A,i}(k_{MI}) + B_q q_{A,i}(k_{MI}) - \frac{\mathbf{1}^T P_{ed,i}(k_{MI})}{V_{dd}}$$

$$q_{B,i}(k_{MI}+1) = C_q q_{A,i}(k_{MI}) + D_q q_{B,i}(k_{MI})$$

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

$$Z_{G,j}(k_{MI}+1) = [Z_{G,j}(k_{MI}) - \mu_{G,j}(k_{MI}) + \varepsilon_{G,j}]^+$$

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

$$Z_G(k_{MI}) \leq Z^{max}$$

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

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*.

3.6.7 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*.

$$\Delta f(k_{SI}) = K_{P,1}[(\lambda_{target}(k_{MI}) - L_{ref}(k_{MI})) - f(k_{SI})] \tag{3.33}$$

$$\Delta L(k_{SI}) = K_{P,2}(L_{ref}(k_{MI}) - L(k_{SI})) \tag{3.34}$$

3.7 Inter-Gateway Management

The optimization problem posed in Section 3.4.2 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 (3.22) 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^i_{kl}, $l \in V_G$.

(ii) Solve the resulting optimization problem over a convex set of continuous variables to find the optimal $\mathbf{r} = \{r^i_{kl}\}$, 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.

3.7.1 Gateway Assignment

In Section 3.4.2 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:

$$
\begin{aligned}
\underset{\mathbf{X}_{assign}}{\text{maximize}} \quad & \min_{j \in V_G} \sum_{i=1}^{N} c_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{i=1}^{N} L_i x_{ij} \leq BW_j, \ \forall j \in \{1,...,M\} \\
& \sum_{j=1}^{M} x_{ij} = 1, \ \forall i \in \{1,...,N\} \\
& x_{ij} \in \{0,1\}, \ \forall i \in \{1,...,N\}, \forall j \in \{1,...,M\}
\end{aligned} \tag{3.35}
$$

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) [122] and many heuristic and exact solution procedures exist [123, 124, 125]. 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 [124] 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.

3.7.2 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.

$$\log MTTF_{ed,i}(\mathbf{r}) = \log \gamma_c + \frac{E_a}{k_B T_{ed,i}(\mathbf{r})} \tag{3.36}$$

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 (3.6), temperature is a function of power dissipation. On the other hand, power is a function of data flow rates through Equation (3.3). 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:

$$T_{ed,i}(\mathbf{r}) = k_1 P_{ed,i}(\mathbf{r}) + k_2 T_{amb,i} + k_3 \tag{3.37}$$

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 (3.37) 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 (3.22), excluding the gateway assignment component can be rewritten as

$$
\begin{aligned}
&\text{maximize} && MTTF \\
&\text{subject to} && \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = L_i,\ i = k\ \forall i,k \in V_{ED} \\
& && \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = 0,\ i \neq k\ \forall i,k \in V_{ED} \\
& && r^i_{kl} \geq 0,\ \forall i,k \in V_{ED},\ \forall l \in S_k \\
& && \sum_{k \in V_{ED}} r^i_{kl} = L_i,\ \forall i \in V_{ED},\ \forall \{l \mid ED_i \in O_l\} \\
& && MTTF \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r^i_{kl} \leq \frac{E_a}{k_B},\ \forall i,k \in V_{ED}
\end{aligned}
\tag{3.38}
$$

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 $y = 1/MTTF$.

$$
\begin{aligned}
&\text{minimize} && y \\
&\text{subject to} && \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = L_i,\ i = k\ \forall i,k \in V_{ED} \\
& && \sum_{l \in S_k} (r^i_{kl} - r^i_{lk}) = 0,\ i \neq k\ \forall i,k \in V_{ED} \\
& && r^i_{kl} \geq 0,\ \forall i,k \in V_{ED},\ \forall l \in S_k \\
& && \sum_{k \in V_{ED}} r^i_{kl} = L_i,\ \forall i \in V_{ED},\ \forall \{l \mid ED_i \in O_l\} \\
& && \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r^i_{kl} \leq y \frac{E_a}{k_B},\ \forall i,k \in V_{ED}
\end{aligned}
\tag{3.39}
$$

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 3.6.1, 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 (3.38) 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 [126]. Under this new objective function, network lifetime optimization is reformulated as a quadratic programming problem.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in V_{ED}} y_i^2 \\
\text{subject to} \quad & \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = L_i,\ i = k\ \forall i,k \in V_{ED} \\
& \sum_{l \in S_k} (r_{kl}^i - r_{lk}^i) = 0,\ i \neq k\ \forall i,k \in V_{ED} \\
& r_{kl}^i \geq 0,\ \forall i,k \in V_{ED},\ \forall l \in S_k \\
& \sum_{k \in V_{ED}} r_{kl}^i = L_i,\ \forall i \in V_{ED},\ \forall \{l \mid ED_i \in O_l\} \\
& \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r_{kl}^i \leq y_k \frac{E_a}{k_B},\ \forall i,k \in V_{ED} \\
& y_i = y_j,\ \forall i \in V_{ED}, \forall j \in S_i
\end{aligned}
\tag{3.40}
$$

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 (3.41), 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.

$$\begin{aligned}
L(\mathbf{y},\mathbf{r},\upsilon,\nu) = &\sum_{i\in V_{ED}} y_i^2 \\
&+ \sum_{k\in V_{ED}} \sum_{\substack{i=1\\ i=k}}^{N} \upsilon_k^i \Big\{ \sum_{l\in S_k} (r_{kl}^i - r_{lk}^i) - L_i \Big\} \\
&+ \sum_{k\in V_{ED}} \sum_{\substack{i=1\\ i\neq k}}^{N} \upsilon_k^i \Big\{ \sum_{l\in S_k} (r_{kl}^i - r_{lk}^i) \Big\} \\
&+ \sum_{i\in V_{ED}} \sum_{j\in S_i} \nu_{ij}(y_i - y_j) \\
= &- \sum_{k\in V_{ED}} \sum_{\substack{i=1\\ i=k}}^{N} \upsilon_k^i L_i + \sum_{k\in V_{ED}} \Big\{ y_k^2 \\
&+ y_k \sum_{l\in S_k} (\nu_{kl} - \nu_{lk}) + \sum_{i=1}^{N} \sum_{l\in S_k} r_{kl}^i (\upsilon_k^i - \upsilon_l^i) \Big\}
\end{aligned} \tag{3.41}$$

The dual function is given by

$$\begin{aligned}
&g(\upsilon,\nu) = \\
&\inf_{\mathbf{r},\mathbf{y}} \left\{ L(\mathbf{y},\mathbf{r},\upsilon,\nu) \;\middle|\; \begin{array}{l} 0 \le r_{kl}^i,\ \forall i \in V_{ED}, \forall l \in S_k \\ \sum\limits_{l\in S_k} E_{kl} \sum\limits_{i=1}^{N} r_{kl}^i \le y_k \frac{E_a}{k_B},\ \forall i,k \in V_{ED} \end{array} \right\}
\end{aligned} \tag{3.42}$$

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 [127, 128] to solve dual problem in a distributed manner.

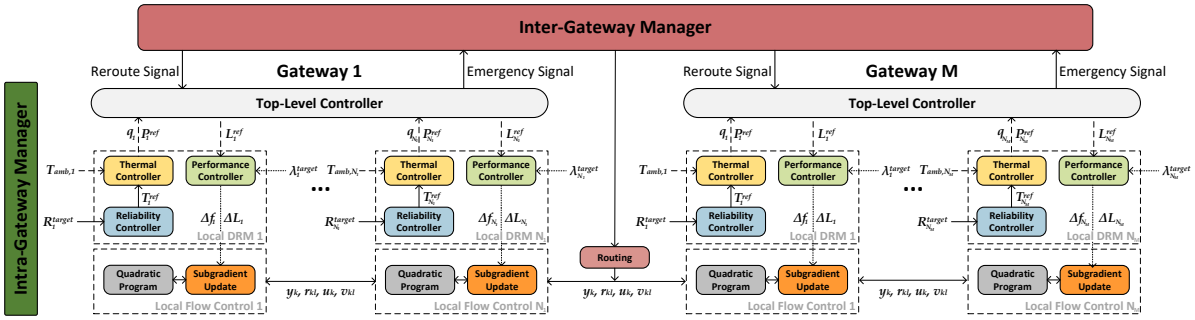


Figure 3.13. Framework block diagram

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 $y_k(t)$, $r^i_{kl}(t)$ for $k \in V_{ED}$, $l \in S_k$.

$$\begin{aligned} \text{minimize} \quad & y_k^2(t) + y_k(t) \sum_{l \in S_k} (\nu_{kl}(t) - \nu_{lk}(t)) \\ & + \sum_{i=1}^{N} \sum_{l \in S_k} r^i_{kl}(t)(\upsilon^i_k(t) - \upsilon^i_l(t)) \\ \text{subject to} \quad & \sum_{l \in S_k} E_{kl} \sum_{i=1}^{N} r^i_{kl}(t) \leq y_k(t) \frac{E_a}{k_B} \\ & r^i_{kl}(t) \geq 0, \ \forall i \in V_{ED}, \ \forall l \in S_k \end{aligned} \tag{3.43}$$

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

$$f^i_k(t) = \begin{cases} L_i - \sum_{l \in S_k} (r^i_{kl}(t) - r^i_{lk}(t)), \ i = k \\ -\sum_{l \in S_k} (r^i_{kl}(t) - r^i_{lk}(t)), \ i \neq k \end{cases}$$

$$h_{kl}(t) = y_l(t) - y_k(t) \tag{3.44}$$

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

$$\upsilon_k^i(t+1) = \upsilon_k^i(t) - \beta(t) f_k^i(t) \tag{3.45}$$

$$\nu_{kl}(t+1) = \nu_k(t) - \beta(t) h_{kl}(t) \tag{3.46}$$

where $\beta(t)$ is a positive scalar step-size. After solving problem (3.43) and updating the Lagrange multipliers, each node exchanges the updated values of y_k, r_{kl}, υ_k, and ν_{kl} with their neighbors $l \in S_k$.

3.8 Evaluation

3.8.1 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. 3.13 shows the block diagram of the simulation infrastructure. Some important model parameters used in the simulation are summarized in Table

Table 3.2. Model Parameters

Parameters	Value	Parameters	Value
a	1.59e-11	g	0.12e+7
b	8.62e-7	SI	0.1 s
C_f	0.22	MI	1 s
C_s	1e-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) [129]. 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* [108] 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 (3.12) 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 [130], 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 [41] 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) [131], implemented on edge devices [92]. 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 [92]. 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).

3.8.2 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 [38]. 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) [45] 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 [10]. Fig. 3.14 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. 3.15 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.

Energy Savings: Fig. 3.16 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

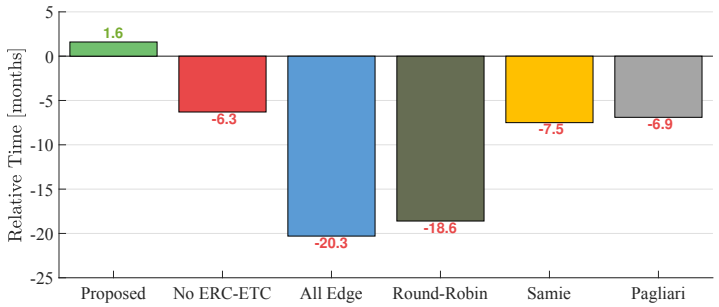


Figure 3.14. The relative time until edge device violates the reliability target of 0.85 (positive is better)

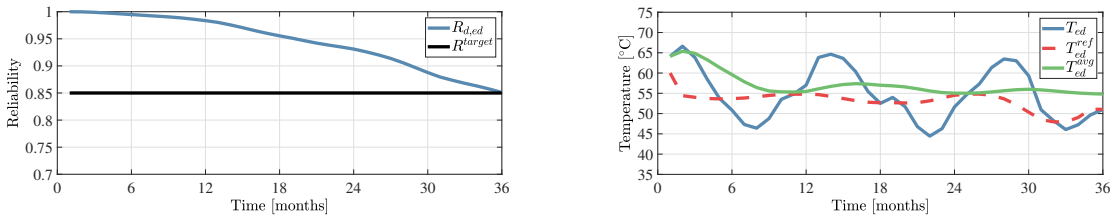


Figure 3.15. Reliability curve and the long time temperature behaviour of the device and the controller

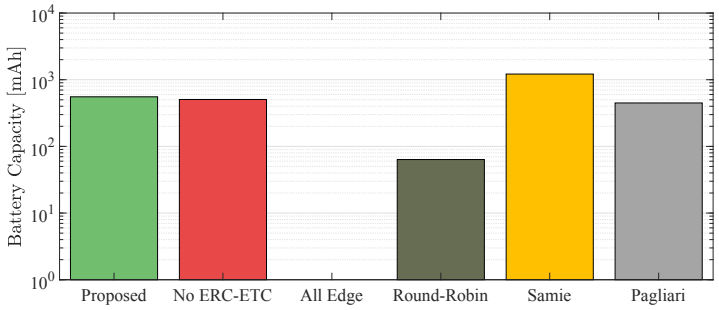


Figure 3.16. Minimum battery charge in the network

of all methods. Our approach shows similar quality in terms of energy efficiency compared to the relative approaches while meeting the reliability requirements. *Samie* [38] 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. 3.14.

Quality of Service: As described in Section 3.4, there are three QoS constraints: input data rate, network bandwidth, and 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.

3.8.3 Multi-Gateway Network Results

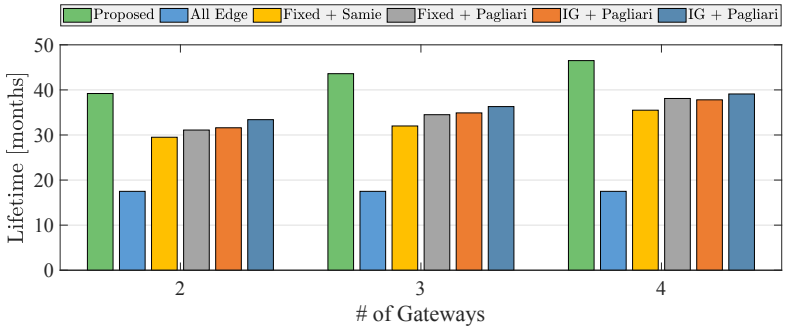
We further evaluate our proposed solution including both the *Intra-Gateway Management* and the *Inter-Gateway Management* components on multi-gateway networks. The experiments are repeated for different number of gateways: 2, 3, 4, and for different number of edge devices: 12, 36. Network devices are 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* [38] and *Pagliari* [45] 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

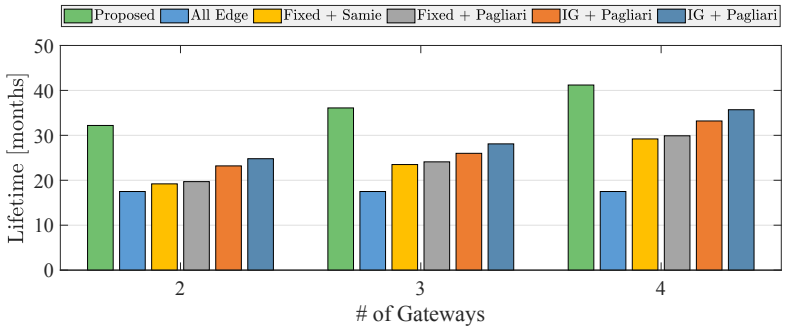
and gateway assignment, denoted by *IG*. This helps us single out and show the contribution of *Intra-Gateway Management* when compared with the *Fixed* versions of the same methods. The evaluated approaches are summarized below.

- *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. 3.17a 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 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.



(a) 12 edge devices



(b) 36 edge devices

Figure 3.17. The lifetime until reliability target violation

For the network with 36 edge devices, as shown in Fig. 3.17b, 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.

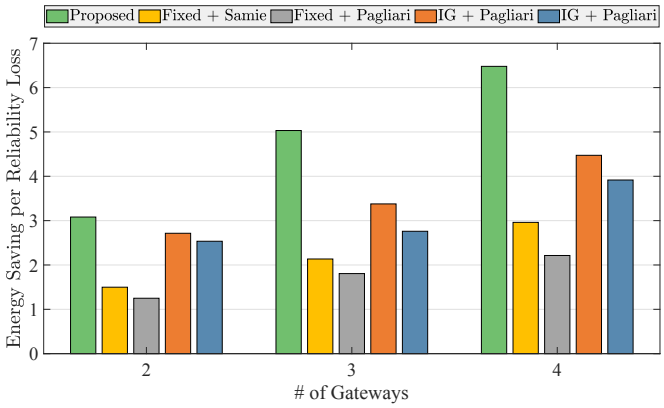
Energy Savings: The primary goal in this work is to reduce maintenance costs of IoT systems by means of reliability management. Maintenance costs arise as a result hardware faults, which require repair, component replacement, or complete node replacement. The hardware faults can be attributed to power outages caused by battery depletion and failures due to reliability degradation. Therefore, as in [132], 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.

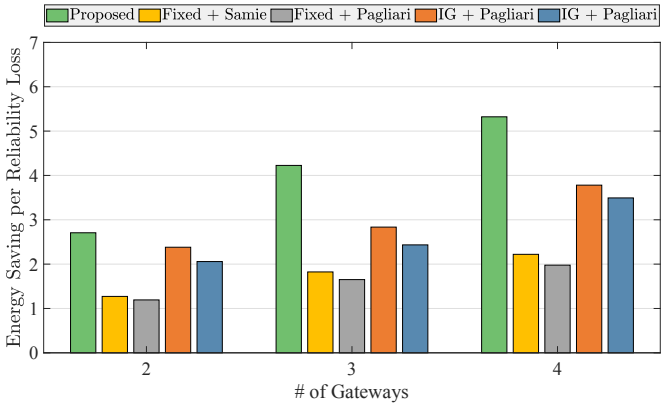
We are interested in an evaluation criteria that covers both energy savings and reliability degradation. In particular, we want to answer the question: "The energy savings come at the cost of how much degradation in reliability?". Intuitively, the amount of loss in reliability should decrease with increasing 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. 3.18a, 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 chapter. 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



(a) 12 edge devices



(b) 36 edge devices

Figure 3.18. Energy savings over reliability degradation

depicted in Fig. 3.18b. 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.

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 3.3. 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* [45] approach considers task execution times which improves its performance in comparison to *Samie* [38].

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

Table 3.3. 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

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 3.4. 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 3.4 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 3.5. Only delay-sensitive tasks are used for these experiments with the aforementioned

specifications.

Table 3.5. 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 3.6. 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:

$$\frac{Mngmt\_Data}{Task\_Data} \times 100\% \tag{3.47}$$

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:

$$\frac{Mngmt\_Msgs}{Task\_Msgs} \times 100\% \tag{3.48}$$

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 [133]. 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 [124]. It should be noted that the frequency of Inter-Gateway Management updates are much lower in comparison.

3.9 Conclusion

In this chapter, 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.

Chapter 3 contains material from "Dynamic Reliability Management of Multi-Gateway IoT Edge Computing Systems", by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in IEEE Internet of Things Journal [2]. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Reliability-Aware Routing in IoT Networks

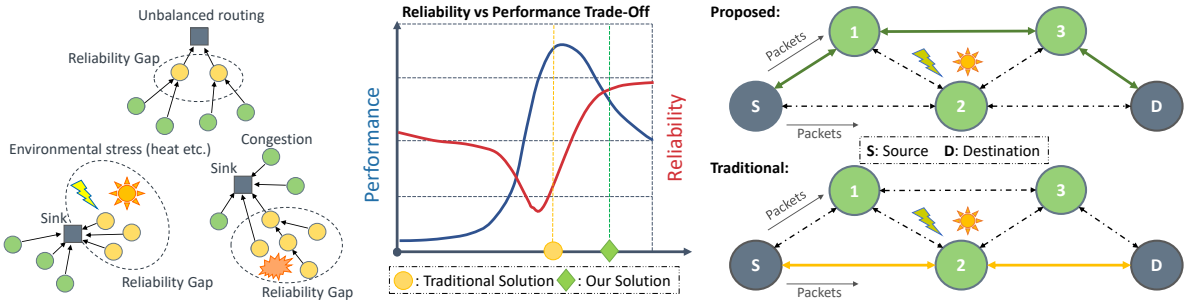
IoT networks operate in diverse and harsh environments that impose thermal stress on IoT devices. The lifetime of these networks can be limited by hardware failures resulting from exacerbated reliability degradation mechanisms at high temperatures. In this chapter, we propose a novel adaptive and distributed reliability-aware routing protocol based on reinforcement learning to mitigate the reliability degradation of IoT devices and improve the network Mean Time to Failure (MTTF). Through routing, we curb the utilization of quickly degrading devices, which helps to lower the device power dissipation and temperature, thus reducing the effect of temperature-driven failure mechanisms. To quantify and optimize networking performance besides reliability, we incorporate Expected Transmission Count (ETX) in our formulations as a measure of communication link quality. Our proposed algorithm adapts routing decisions based on the current reliability status of the devices, the amount of degradation they are likely to experience due to communication activity, and network performance goals. We extend the ns-3 network simulator to support our reliability models and evaluate the routing performance by comparing with state-of-the-art approaches. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance.

4.1 Introduction

The Internet of Things (IoT) continues to rapidly develop as it is adopted progressively across many domains such as logistics, farming, industrial and environmental monitoring, healthcare, and smart infrastructures. The number of interconnected IoT devices will reach 40 billion by 2025 and worldwide spending on the IoT is already more than $750 billion [8]. Unfortunately, coupled with such dramatic growth, the inherent large-scale of the IoT brings a maintainability challenge with high costs. Currently, the operational expenses reach up to 80% of the overall cost [7], of which a significant fraction is due to hardware failures. While meeting the needs of a growing range of applications, it is also a crucial requirement for IoT devices and networks to operate reliably for long periods, otherwise, maintenance investments can become a critical bottleneck for the growth of IoT.

Recent advances in energy harvesting techniques combined with energy-efficient approaches at different layers of the networking stack made it possible for IoT devices to have substantially prolonged battery lifetimes. With batteries continuously being recharged by energy harvesting sources, energy-neutral operation [134] for the network can be ensured. In such networks, since the risk of batteries running out of energy is diminished, the limiting factor for network lifetime are hardware failures due to reliability issues. As a result of aging and degradation, components in IoT devices lose reliability and eventually fail, leading to a permanent loss of functionality.

Previous research has shown that reliability degradation of electronics worsens exponentially with increasing temperature due to intensified effects of various mechanisms such as Time-Dependent Dielectric Breakdown (TDDB), Electromigration (EM), Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) [11, 55, 12]. IoT devices are often deployed in harsh environments, resulting in stress on the hardware to reduce their reliability and mean time to failure (MTTF). The majority of them typically do not have active cooling to mitigate the thermal stress. In such cases, curbing power dissipation of devices helps to lower the device



(a) Reliability problems **(b)** Our solution vs traditional solutions

Figure 4.1. Reliability-driven routing in IoT networks.

temperatures and scaling down the effect of temperature-driven failure mechanisms to achieve a better MTTF. Network routing can be useful in this regard; it is possible to place the IoT devices into low-power states by avoiding them in communication paths. In this way, low reliability devices in the network are utilized less to reduce thermal stress and slow down degradation.

Many energy-based routing algorithms have been proposed [135, 136, 137] with the goal of extending the battery lifetime of IoT networks, but no work considers the reliability of IoT devices. Here we refer specifically to the aging and reliability degradation of the hardware of an IoT device, not the communication reliability or soft errors that are broadly studied. The literature on network routing does not scrutinize the problem of hardware failures and reliability issues as a bottleneck for the lifetime of networks. To improve the MTTF of IoT networks, a reliability-aware routing should be designed following these principles:

1) Avoid the weakest nodes: As shown in Figure 4.1a, there are many situations that may result in an unbalanced reliability degradation in IoT networks. We call this phenomenon *reliability gap*, the situation in which there is a significant reliability difference between different nodes of the network. Reliability gap can arise as a result of convergecast traffic patterns, congestion, environmental stress, and disproportionate communication distances (Figure 4.1a). For networks with convergecast traffic patterns and local congestions, some regions in the network may have more traffic to forward, and hence, the nodes here are active more often than the others. Similarly, some nodes may be exposed to higher thermal stress due to their physical

location, especially in applications such as industrial and environmental monitoring. These nodes may be the bottleneck for network lifetime since their reliability degrades rapidly and will be the first to fail because of shortened MTTF. The networking load should be distributed as a function of the reliability state of the nodes–besides network performance–by using reliability-aware routing, thus avoiding the weakest nodes.

2) Use efficient communication links: If the communication link is of low quality and inefficient, then the transmitter node must send many copies of the same packet to be correctly captured by the receiver. Multiple retransmissions mean that the transmitter and receiver will stay active and experience reliability degradation until a successful reception takes place. Using better links improves both communication performance and device reliability. Therefore, the routing protocol should be aware of the link quality and its influence on device reliability.

The typical approach for routing is to model the network as a weighted directed graph and then find paths with the minimum cumulative weight. The weights of the graph edges and vertices traditionally include a variety of node and link metrics: latency, hop count, stability, bandwidth, throughput, and energy, or may be a composite of multiple metrics [58, 59]. The impact of degradation mechanisms on device reliability has not been taken into account by routing techniques to date. A simple example of how reliability might affect routing is shown in Figure 4.1b. Node 2 is in a location exposed to higher temperature and thus has much higher thermal stress than the other nodes. A traditional routing solution, shown in yellow, in the bottom right figure, selects the least hop path between the source (S) and the destination (D) nodes, routing through node 2, thus causing its early failure. Our solution, shown in the top right figure, in green, takes a slightly longer path through nodes 1 and 3 but avoids the early failure of node 2. Performance and reliability trade-off over a continuous sample space of possible routing strategies is shown on the left of Figure 4.1b. Traditional routing solutions purely aim at maximizing the aforementioned performance-related metrics. In contrast, the goal of a reliability-aware routing solution is to find a favorable middle ground between performance and reliability.

In recent years, the complexity, dynamism, and heterogeneity of modern IoT networks have driven a recent development of routing techniques based on reinforcement learning (RL)[138, 139]. Traditional routing techniques, which are based on statistical assumptions regarding traffic flows and network conditions, are more and more perceived as inefficient to suit the diverse, complex, and highly changing conditions of IoT networks. RL-based routing techniques have been shown to successfully address these challenges; they automatically learn the dynamics of networks, such as new flow arrivals, congestion points, topology changes, quality of links, and adapt to it.

In this chapter, we propose R3-IoT, a distributed reinforcement learning based reliability-aware routing protocol for IoT networks. We maximize the reliability and hence the MTTF of the most degraded nodes by (i) avoiding them in the communication path to minimize their traffic, (ii) using high quality, reliable links to reduce retransmissions. Reliability and Expected Transmission Count (ETX) [140] metrics are incorporated in the reinforcement learning formulation. The routing policy learns from experience at runtime, using the past routing decisions and their outcomes, to achieve high performance and to prolong the network lifetime. To the best of our knowledge, we are the first to present a routing solution that explicitly addresses the reliability degradation problem in IoT networks. We conduct extensive simulations using a real-world ambient temperature dataset from the National Solar Radiation Database (NSRDB) [141]. Our evaluation uses large-scale sensor network data, along with real device measurements and models from the High-Performance Wireless Research and Education Network (HPWREN) [76]. Since reliability is difficult to evaluate in practice, we use simulations based on an extended version of ns-3 [17] to demonstrate that our routing approach can achieve similar performance compared to the state-of-the-art while showing up to 73.2% improvement in reliability for various communication data rates and number of nodes in the network.

4.2 Related Work

4.2.1 Reliability in Routing

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 categorized into three groups, none of which handle hard errors: soft errors (causing random bit flips) [50], software reliability issues [51], or batteries running out of energy [52, 53]. For example, in [51], 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 [54]), 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. We propose a routing solution that explicitly addresses the reliability degradation due to hardware failure mechanisms, which is different from previous works.

Hard failures, caused by well-known thermally-driven mechanisms in silicon, such as TDDB, EM, BTI, result in a need to replace that electronic component in the field, leading to high maintenance and replacement costs. Hard failure models have been studied extensively at the circuit and chip level [11, 55, 12], and adopted for dynamic voltage & frequency scaling, task scheduling, and power gating strategies in multi-core system-on-a-chips [10]. Prior to our work in [142], nobody has considered how hard failures affect problems at the network level and how networking might affect the electronics reliability. This work extends and improves [142] by introducing a new reinforcement learning based protocol, whereas only a routing metric was proposed in the former.

4.2.2 Maximum Lifetime Routing

The problem of maximum lifetime routing has been extensively researched over the last two decades for Wireless Sensor Networks (WSNs) and more recently for IoT networks. Since the majority of WSN and IoT devices are battery-operated, the works in this domain are directed towards improving battery lifetimes. In contrast to the approaches that aim at minimizing the total or average energy consumption (e.g., LEACH [143], GEAR [144], ER-RPL [137]), maximum lifetime routing can ensure a balanced depletion of energy among the network nodes. By incorporating the residual energy of node batteries into routing decisions, quickly draining nodes are avoided in the communication paths, and hence network lifetime is extended. A detailed survey on this topic can be found in [52, 56, 57]. We next discuss a few representative publications.

Chang and Tassiulas [53] define the communication link cost as a function of remaining node energy and the required transmission energy for using that link. By using the Bellman–Ford shortest path algorithm for the computed link costs, the least cost path – whose residual energy is the largest – is found. Following similar methodologies, metrics such as link quality [145], throughput [146], queue utilization [147], and so on were combined with residual energy to achieve different objectives along with the network lifetime. The authors of [148] and [149] proposed evolutionary algorithms for balancing the load and energy consumption of the nodes. In [150], the traffic load is estimated and the optimal data path is computed to avoid energy holes by efficiently utilizing the nodes that are susceptible to congestion. Although a myriad of studies analyzed the network lifetime problem from an energy optimization perspective, to date, there is no work that addresses the reliability issues in the way we do in this work.

Only a small subset of the proposed routing algorithms have found application in practice. There are prevalent routing protocols that have gone through the standardization process, which demands a lot of time, effort, and is expensive. Thus, if impact is needed, building a protocol completely from scratch is undesirable. Following this philosophy, modifications and

improvements to various standardized protocols such as AODV [151], OLSR [152], RPL [153] were proposed with a focus on extending network lifetime. Many studies engineered new routing metrics that take into account residual battery energies [154, 155, 156]. Further modifications are proposed in [157, 117]. Similarly, in our work we adopt and build upon the AODV (Ad hoc On-Demand Distance Vector) protocol.

4.2.3 Reinforcement Learning Based Routing

Reinforcement learning based routing is gaining importance with the ever-increasing complexity and dynamism of IoT networks and the recent advancements in machine learning [138]. The first application of RL in routing by Boyan and Littman [158] demonstrated that RL is indeed a promising solution for complex communication networks. Since then, many studies have been conducted using variants of RL algorithms with different networking objectives and requirements. Most of the works in RL-based routing have utilized the well-established Q-routing algorithm [158] as their underlying idea, albeit with some improvement [159, 160, 161]. Q-routing is based on the traditional Q-learning model in which each node makes its routing decision based on the local routing information. Among many different objectives employed using Q-routing based algorithms, the work in [161] reduces end-to-end delay. Work in [159] couples Q-routing with on-policy Monte Carlo to reduce energy consumption and enhance the network lifetime. The authors of [160] introduce a dynamic discount factor to Q-learning for reducing the amount of route discovery processes after a link failure occurs.

A few recent works consider a combination of reinforcement learning with AODV routing. For example, Q-learning AODV (QLAODV) [162] is a routing protocol that considers link stability and bandwidth efficiency. In [163] the authors use a Bayesian Network to estimate congestion levels and tune the learning weights where they consider signal to noise ratio, delay, and throughput for making routing decisions. Residual battery levels and energy efficiency were also explored as routing objectives in [164], where they are utilized to adjust the willingness of nodes to participate in AODV routing with the SARSA learning algorithm. In contrast to the

previous works on RL based routing, we consider the reliability of network devices which was not studied before and we try to maximize hardware lifetime. Reinforcement learning based AODV was particularly studied in MANETs and VANETs due to their erratic mobility, energy consumption, and traffic profiles. It has shown promising results because of its adaptability in such highly dynamic network conditions. Following this rationale, we propose a novel distributed Q-learning based adaptive AODV routing approach. In our work, we model the dynamic factors such as ambient temperature and computation workloads of IoT devices, as well as their effect on reliability. We assume that IoT devices run various workloads which contribute to their heating, combined with the thermal stress imposed by the environment. Our proposed approach is able to adapt these variations in the network and discover better routes without having to know the network topology and traffic patterns in advance.

To summarize, our main contributions are as follows:

- We explicitly consider hardware reliability in IoT network routing to reduce failures and improve network lifetime.
- We incorporate node reliability and ETX metrics into Q-learning updates in our reinforcement learning based approach. Through ETX, we assess the expected communication link performance as well as the expected reliability degradation of a node. Thus, routing decisions are driven by the current reliability of the nodes on the path, amount of degradation they will experience due to retransmissions, and networking performance.
- We implement our routing mechanism in a novel routing protocol called R3-IoT.
- We model reliability in the ns-3 network simulator. To accomplish this, we include the ambient temperature and computation workloads of IoT devices in our simulations.
- We compare R3-IoT to state-of-the-art routing protocols and show that the network reliability is significantly improved while achieving similar performance.

4.3 Reliability Modeling and Simulation

4.3.1 Device Modeling

Reliability is defined as the probability of not having failures up to a given time t. The reliability function $R(t)$, in general, can be expressed as a function of *failure rate* $\lambda_f(t)$ [55]:

$$R(t) = e^{-\int_0^t \lambda_f(t')dt'} \tag{4.1}$$

From Equation (4.1), the rate of how quickly reliability is degrading is determined by the failure rate, which depends on temperature, aging, device power state, and switching frequency between power states.

In our reliability analysis, we focus on hard failure mechanisms that cause irrecoverable device failures. Mechanisms such as Time-Dependent Dielectric Breakdown (TDDB), Electromigration (EM), Bias Temperature Instability (BTI), and Hot Carrier Injection (HCI) induce reliability degradation, and thus eventually cause failures. Failure rate models have been developed for each mechanism, which show an exponential dependence on temperature that can be described as follows:

$$\lambda_q = A_0 \eta_q e^{-\frac{E_a}{kT_q}} \tag{4.2}$$

$$\forall q \in \{Active, Idle, Sleep, ...$$

$$TransitionToSleep, TransitionToActive\}$$

where A_0 is an empirically determined constant, E_a is the activation energy, k is the Boltzmann's constant, and η_q is a constant depending on the respective mechanism and device. Here, we consider temperature T_q of a device as a function of its power state, ambient temperature, and time. When a device switches to a different power state, temperature increases/decreases until it converges to a new steady-state value after a certain time. We assume that the IoT devices

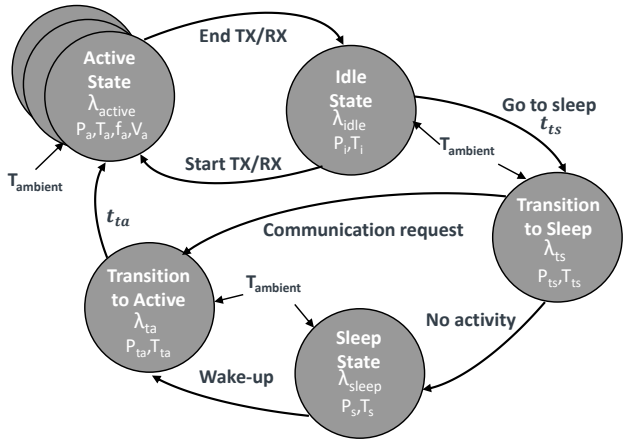
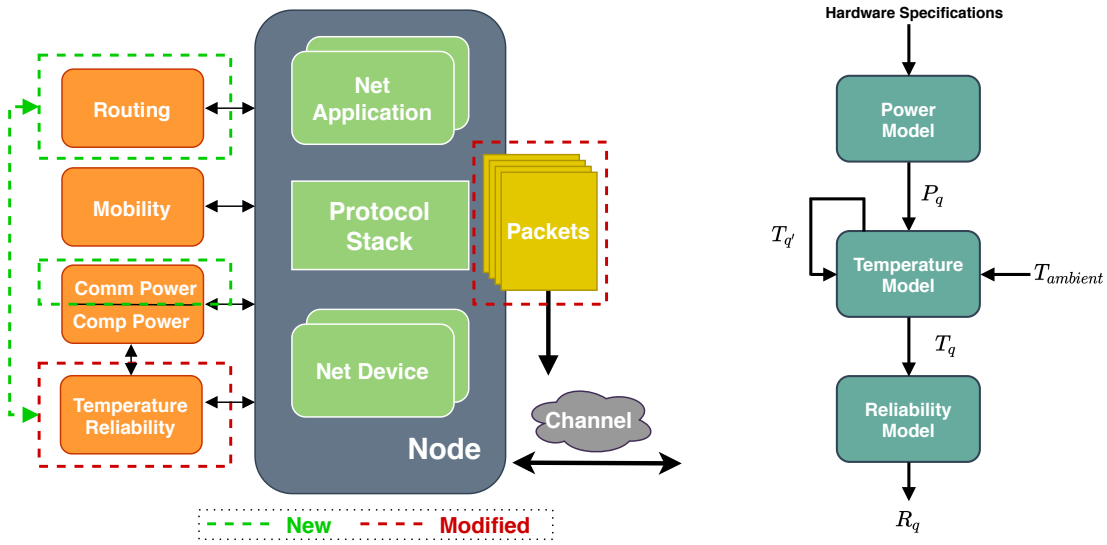


Figure 4.2. Device state model.

can be in various operational states (e.g., *Active*, *Idle* etc.) denoted q, characterized by power dissipation, voltage, and frequency.

Figure 4.2 depicts a sample power state diagram of the device with the state transition mechanisms and the parameters characterizing the states. Such state-based model is able to represent the dynamics of IoT devices for many applications, yet convenient and adaptable for simulation purposes. For IoT devices, switching between power states using duty-cycling and wake-up radio techniques – usually implemented at Medium Access Control (MAC) layer – is common for energy saving purposes [165]. The objective of state transitions (represented with arcs) is to put the IoT device in low power modes when not communicating. In the idle state, the system-on-a-chip (SoC) of the device is powered on but not communicating or processing any packets. In the sleep state, most of the SoC subsystems are power-gated. In the active state, the device is busy transmitting/receiving and processing packets. Power scaling methods such as DVS policies can be used for transition between active states for some IoT devices, either down-scaling to reduce power consumption or up-scaling to meet an application performance criteria [134]. Transition to sleep and transition to active states model the time and power consumption required to enter and exit the sleep state. Transition times to/from low-power states follow average transition times t_{ts}, t_{ta}, respectively. Failure rates and the amount of induced



(a) ns-3 node structure

(b) Reliability computation

Figure 4.3. Reliability simulation in ns-3.

reliability degradation change with each power state since different levels of power consumption result in different temperature profiles. Ambient temperature heavily influences the device's internal temperature and has an effect in every operational state.

4.3.2 Reliability Simulation for IoT Networks

Analytical models for power, temperature, and reliability should be used to enable reliability evaluation and analysis in network simulations. In this work, we leverage the recently proposed *RelIoT* [166] framework for the ns-3 simulator and enhance it according to our reliability modeling discussion.

The original framework offers an application-based power model that characterizes power consumption of different applications running on IoT devices, particularly targeting edge computing scenarios. Figure 4.3a depicts the node structure in ns-3 augmented with the *RelIoT* framework, our modifications, and new additions. We implement a new power state machine model as in Figure 4.2, for the communication component of a node's power consumption module. The state transitions take place according to the node's communication activity. To compute reliability, power/temperature/reliability model flow is initiated by state transitions

as shown in Figure 4.3b. We adopt *RelIoT*'s first order differential temperature model, which incorporates the dependence of node's internal temperature on power and ambient temperature. However, we change the temperature update mechanism. The model dynamically updates device temperature when ambient temperature changes or a state transition occurs. During the transient period, temperature increases/decreases until converging to the steady-state temperature of the new operational state. Our modified reliability model dynamically updates the node's reliability through Equation (4.3) by recursively subtracting the degradation induced between consecutive state transitions. The current implementation uses the reliability model presented in [12].

$$R_q = R_{q'} - \Big(\underbrace{R(t_{q'}, T_q) - R(t_q, T_q)}_{\text{degradation}} \Big) \tag{4.3}$$

The subscripts q and $q^{'}$ indicate the current and previous states respectively. T_q is the temperature experienced by the device between two state transitions from time instants $t_{q'}$ to t_q. $R(\cdot)$ is the static reliability function described in Equation (4.1). Finally, the reliability model connects with our routing protocol so that reliability values can be monitored by the routing algorithm. Packet structures are modified to accommodate for the requirements of our learning algorithm, which are explained into more detail in Section 5.

In real systems, reliability tracking is possible with degradation, stress, or aging monitors [167, 34, 35]. These monitors, based on ring oscillators that convert temperature and voltage stress into oscillation frequency, give information on the accumulated stress of the SoC, which is used to estimate failure rates and reliability degradation caused by different mechanisms (e.g., TDDB, EM, NBTI) [167, 34]. As an example use for IoT, the authors in [35] implement an on-chip stress monitor for IoT devices and pave the way for its future usage in IoT maintenance and management. The system reliability degradation status is usually an input into dynamic reliability management (DRM) techniques [168] to improve device usage. In our case, we propose to use reliability status as an input to our network routing protocol to drive routing

decisions.

4.4 Reliability-Aware Routing with Reinforcement Learning

In the following, we first describe the system model and the problem. We next present our proposed reinforcement learning based routing algorithm.

4.4.1 System Model

Consider an ad-hoc wireless IoT network deployed in a mesh topology. The network consists of heterogeneous nodes that we classify into three categories: sensors, gateways, and servers. Each category can have devices of different types. For example, the sensor node can have ARM Cortex-M4 or ARM Cortex-A7 processors. The sensor nodes are 'source' nodes that generate data, which needs to be communicated to a sink (i.e. server). Since the network topology is mesh, nodes can cooperate to distribute and relay data in a multi-hop fashion. We assume that gateways can do data processing.

We investigate a network with many sensor and gateway nodes but a single sink node, similar to related works [162, 163, 164]. There are a total of $N+1$ nodes, where nodes $i = 1, ..., N$ denote sensor and gateway nodes and $i = N+1$ denotes the sink node. Source (sensor) nodes generate data at rates r_i, so the sink node receives their sum $\sum r_i$. The distance between nodes i and j is $d_{i,j}$. Let N_i denote the set of neighboring nodes to which node i can send packets to. Then, $N_i = \{j : d_{i,j} < d_{max}\}$, where d_{max} is the distance of transmission with maximum power. The notation $j \in N_i$ is used to show that node j is a neighbor of node i and they can communicate. The nodes are static with fixed distances, so neighbors do not change. However, the communication between neighbors is not perfect at all times because we assume lossy communication links, which cause random packet drops.

As shown in Figure 4.4, nodes are characterized by their power consumption, temperature, reliability, and the ambient temperature around them. We assume an energy harvesting network, so the residual energy of batteries is not included in our model. In a heterogeneous network,

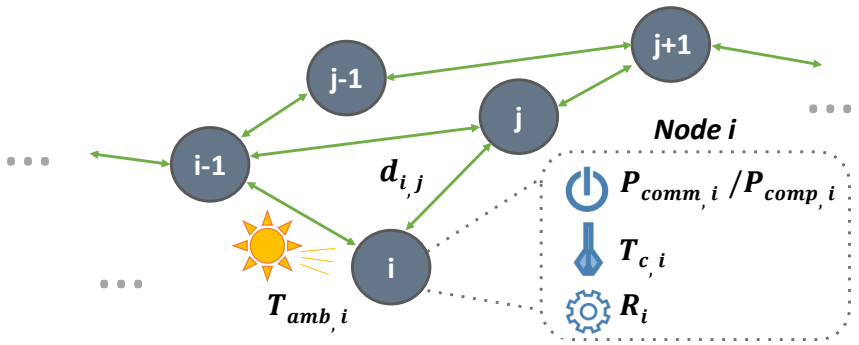


Figure 4.4. System model.

the power, thermal, and reliability characteristics vary between different nodes. Each node consumes the power amount $P_{comp,i}$ for computation and $P_{comm,i}$ for communication. Even though our routing approach only has an impact on communication power consumption, the routing decisions are made by taking into account the overall device temperature and reliability. IoT devices usually run various workloads throughout their operation that contribute to their power consumption. Therefore, we also consider the computation power consumption for the nodes. The core temperature $T_{c,i}$ of the nodes is influenced by their overall power dissipation $P_{comm,i} + P_{comp,i}$ and by the ambient temperature $T_{amb,i}$. Finally, R_i denotes the reliability, which is heavily affected by temperature $T_{c,i}$. For the routing algorithm, the computation power consumption $P_{comp,i}$ and ambient temperature $T_{amb,i}$ are 'external' factors that influence node reliabilities. A routing algorithm cannot control these variables, but the routing decisions should be made in consideration of their effect on the overall reliability.

In such a setting, we seek to improve the network's mean time to failure (MTTF) by means of routing, while keeping performance at adequate levels. MTTF of a single node can be expressed through Equation (4.4) as a function of reliability:

$$MTTF = \int_0^\infty R(t)dt \tag{4.4}$$

where $R(t)$ is the reliability if a device at time t. Then, network MTTF is the minimum of any

node in the network (i.e. $\min_{i \in N} MTTF_i$). Here we assume that the network lifetime is defined as the time of first node's failure, which is a common assumption. This definition is one of the most prevelant in literature [52] and was used in many recent works [117, 118]. Improving the network's MTTF, or its lifetime in general, can be accomplished by maximizing the reliability of the weakest nodes in the network and minimizing the overall reliability degradation on the nodes. If the weakest node in the network has high reliability, then the time it takes for the first node to fail on average will be extended. As stated previously, how quickly a node's reliability degrades relates to its communication activity, which is influenced by routing decisions. Hence, routing can help prolong network lifetime.

Our goal is to have a protocol that improves the reliability, and hence the MTTF of the most degraded nodes, as well as takes into account the performance in its routing decisions. The routing protocol should (i) help avoiding paths with the minimum reliability nodes, (ii) utilize efficient communication links, and (iii) lead to decisions that will induce minimal reliability degradation.

4.4.2 Reinforcement Learning: Background

Reinforcement learning (RL) [169] is a framework in which an *agent* learns control policies based on experience, for making decisions in an *environment*, to optimize a given notion of rewards. The agent interacts over time with its environment, collects information, and selects the *action* to be applied according to its goal and current state. What is good or what is bad for the agent is defined by the *reward signal*. The environment returns a reward to the agent on each time step to provide feedback about the effect of the recent taken action. The total amount of reward an agent can expect to accumulate, starting from the current state, is estimated by the *value function*. Reward signals indicate what is good (or bad) in an immediate sense, whereas value functions indicate what is good (or bad) in the long run. Therefore, usually value functions are the primary criteria when making and evaluating action decisions.

The problem of reinforcement learning is mathematically framed using a 4-tuple $(\mathscr{S}, \mathscr{A},$

$\mathscr{P}, \mathscr{R}$), where $\mathscr{S}$ is the set of states, $\mathscr{A}$ the set of actions, $\mathscr{P}$ the state transition probabilities, and $\mathscr{R}$ the rewards. The agent and environment interact at discrete time steps (also called epochs), $t = \{0, 1, 2, 3, ...\}$. At each time step t, the agent observes state $s_t \in \mathscr{S}$, then selects an action $a_t \in \mathscr{A}$. As a consequence of its action, one time step later, the agent receives a reward, $r_{t+1} \in \mathscr{R}$, and finds itself in a new state, s_{t+1}. The probability of moving from state s to state s' by taking action a is:

$$p(s'|s,a) = Pr\{s_{t+1} = s'|s_t = s, a_t = a\}$$
$$\sum_{s' \in \mathscr{S}} p(s'|s,a) = 1 \tag{4.5}$$

The objective of the agent while selecting a certain action is to maximize the total cumulative reward it receives over its lifetime. Thus, the agent should learn which of its actions are desirable in the long run. Based on this, actions are selected such that the *return* G_t, expressed as the sum of discounted rewards, is maximized:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4.6}$$

where $\gamma \in [0, 1]$ is the discount factor. If $\gamma = 0$, the agent is called 'myopic', only being concerned with maximizing the immediate reward. As γ approaches 1, the future rewards are taken into account more.

The rewards the agent can expect to receive in the future depend on which actions it will take. A *policy* π defines how the agent selects its actions. Formally, the policy $\pi(s,a)$: $\mathscr{S} \times \mathscr{A} \mapsto [0, 1]$ is a mapping from states to probabilities of selecting each possible action. The value function $V^{\pi}(s)$ is the expected return of following a policy π when at a state s:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|s_t = s] = \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\Big] \tag{4.7}$$

Here, $\mathbb{E}_\pi[\cdot]$ denotes the expected value if the agent follows the policy π. The function V^π is particularly called as the *state-value* function. A similar function, denoted $Q^\pi(s,a)$, is defined for the expected return of starting from state s and taking action a under policy π. It is called as the *action-value* function and expressed as: $Q^\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$.

The goal of reinforcement learning is to find a policy that results in maximum reward in the long run. A policy π is better than a policy π' if its expected return is greater than that of π' for all states, i.e., $\pi > \pi' \Leftrightarrow V^\pi(s) > V^{\pi'}(s), \forall s \in \mathscr{S}$. The *optimal policy* π^* has the optimal state-value function V^* and action-value function Q^*, given as follows:

$$\begin{aligned} V^*(s) &= \max_\pi V^\pi(s), \ \forall s \in \mathscr{S} \\ Q^*(s,a) &= \max_\pi Q^\pi(s,a), \ \forall s \in \mathscr{S}, \forall a \in \mathscr{A}(s) \end{aligned} \tag{4.8}$$

The optimal value function $V^*(s)$ can be expressed through Equation (4.9) without reference to any policy, in a special form called the *Bellman equation* [169].

$$V^*(s) = \max_{a \in \mathscr{A}} \left[r + \gamma \sum_{s' \in \mathscr{S}} p(s'|s,a) V^*(s') \right] \tag{4.9}$$

Once V^* is obtained, the optimal policy can be determined by solving a system of equations. However, explicitly solving the Bellman equation is rarely practical due to the large solution space present in the majority of RL problems. There are many different methods that approximately solve the Bellman equation at reasonable computational cost. In our problem, we use such a method –Q-learning– since we deal with large networks and the learning has to take place on resource-constrained IoT devices.

4.4.3 Q-Learning Based Routing Algorithm Design

In this work, we use Q-learning for routing, which is a model-free off-policy temporal difference reinforcement learning approach [169]. Q-learning is based on the value of state-action

pairs $Q(s,a)$, called *action-value* function. We define the value of taking action a in state s under a policy π, $Q^{\pi}(s,a)$ as the expected return from taking such an action and thereafter following policy π:

$$\begin{aligned} Q^{\pi}(s,a) &= \mathbb{E}_{\pi}[G_t|s_t = s, a_t = a] \\ &= \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\Big] \\ &= r + \gamma \sum_{s' \in \mathscr{S}} p(s'|s,a) V^{\pi}(s') \end{aligned} \tag{4.10}$$

Thus, from Equation (4.9) and Equation (4.10), we have:

$$V^*(s) = \max_{a \in \mathscr{A}} Q^*(s,a) \tag{4.11}$$

In Q-learning, the agent learning consists in a sequence of stages, called epochs. In epoch t, the agent is in state s_t, it performs action a_n, it receives a reward r_t, and it moves to state s_{t+1}. The action value is updated as follows:

$$Q(s_t,a_t) \leftarrow (1-\alpha)Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a \in \mathscr{A}} Q(s_{t+1},a_{t+1})] \tag{4.12}$$

where α is the learning rate. The agent learns optimal policy with the help of a greedy policy where an action can be chosen just by taking the one with the maximum Q-value for the current state. Q-learning is a suitable method for online 'runtime problems' such as routing in our case, because it only uses the most recent decisions to update its policy. Also, it converges to the optimum action-values with probability 1 as long as all actions can be randomly sampled in all states [170]. For this reason, it is an effective technique for learning from delayed reinforcement, i.e. learning based on the reward that can be received far in the future. To map a routing problem to a Q-learning problem, one needs to design the state space, action space, reward function, and learning parameters:

- *State (s):* The current state of the agent is the index of the node holding the packet.
- *Action (a):* Selection of the node for the next hop.
- *Reward (r):* Higher rewards earned if the action brings the packets closer to the destination node.
- *Learning Parameters (γ,α):* Described in terms of reliability and performance related metrics of the nodes.

States and Actions

For learning a routing strategy, each node is associated with a state s, and for each of its neighbor s', there is a corresponding action. Executing action a at s means forwarding the packet to the neighbor s' corresponding to that action. Let $s = i$ denote the node holding a packet P to forward and $Q_i(des, j)$ denote the Q-value of node i forwarding the packet to destination d through next-hop node $s' = j$. Then, the action-value updates are expressed through Equation (4.13).

$$Q_i(des, j) = (1-\alpha)Q_i(des, j) + \alpha[r + \gamma \max_{k \in N_j} Q_j(des, k)] \tag{4.13}$$

Node i maintains a table of Q-values $Q_i(des, j)$ for each neighbor j and destination des, which can be regarded as its routing table, telling which neighbor to forward the data. With the help of this routing table, the optimal routing path can be constructed by a sequence of table look-up operations. We treat the network as the environment and the nodes as the entities where the agents reside. For a completely distributed Q-learning based routing, we assume that there exists an agent at each node. The agents try to improve the current solution while switching between exploration and exploitation of the solution space. The exploration and exploitation processes of Q-learning have their own interpretations for routing. In our approach we adopt the $\varepsilon - greedy$ strategy. By default, a node selects the next hop node which has maximal Q-value to forward data, which is called exploitation. However, with some probability ε it chooses a random node which does not have maximal Q-value, this is called exploration.

Reward Function

The reward function is critical to Q-learning, as it determines the behavior and performance of the agent. The goal of the routing algorithm employing Q-learning is to get the packet delivered to the destination (the sink node) with maximum expected return, i.e., minimum cost. To prevent loops and ensure forwarding the packets towards the destination node, we need increasing Q-values in the direction of the destination. Hence, we employ the following reward function:

$$r = \begin{cases} 1 & \text{if } i \in N_{des} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

where N_{des} is the set of neighbors of the destination des. This means that if a node receives a packet from the destination, the reward will be 1 and otherwise 0. Therefore, only the immediate neighbors of the sink node receives a reward of 1. By using this reward function, closer nodes to the destination attain high Q-values. This reward propagates to the nodes away from the destination, but it gets discounted more and more as it travels further.

Reliability-Aware Learning Parameters

To include both reliability and performance aspects in our Q-value updates, we propose a composite discount factor (γ) function of a node's *reliability* and its *expected transmission count* (ETX) over communication links. Using this approach, Q-value of a node decreases at each update if its reliability and link quality is low. The optimal policy is then the one which picks both reliable nodes and reliable communication links for routing.

ETX is one of the most frequently used metrics in routing protocols. It estimates the number of data transmissions required to send a packet over a link and get acknowledged, including retransmissions. It is computed as:

$$ETX = 1/(p_{i \mapsto j} \cdot p_{j \mapsto i}) \tag{4.15}$$

Here, the notation $p_{i\mapsto j}$ denotes the probability of successful packet delivery from packet source node i to destination node j. We obtain both probabilities $p_{i\mapsto j}$ and $p_{j\mapsto i}$ with a message exchange protocol that is explained in Section 5. Through ETX, we assess the link performance as well as the expected reliability degradation for using that link. To calculate the reliability degradation induced, we first estimate the traffic that the node has to forward. Let j denote the node of interest, then the total allocated traffic for j is the sum of traffic it generates TR_j^{gen} and the traffic TR_i^{total} incoming from its neighbors i:

$$TR_j^{total} = TR_j^{gen} + \sum_{i|i\mapsto j} TR_i^{total} \tag{4.16}$$

Multiplying this estimate of total traffic TR_j^{total} with the expected transmission count $ETX_{j\mapsto i}$ and then dividing by the data rate r_j of the node, we compute the expected communication time through Equation (10).

$$t_j^{comm} = (TR_j^{total} \cdot ETX_{j\mapsto i})/r_j \tag{4.17}$$

Finally, in Equation (11), we estimate the reliability degradation D_j induced by using Equations (4.1) and (4.3).

$$D_j = R(t_{s,j}, T_{s,j}) - R(t_{s,j} + t_j^{comm}, T_{s,j}) \tag{4.18}$$

Since D is proportional to ETX, its value will be higher for low-quality links. Selecting paths with the minimum D utilizes efficient links (better performance with fewer packet loss) and induces minimal reliability degradation.

As the second part of our composite function, we need a component that focuses on the bottleneck in network MTTF: the node with minimum reliability. Thus, the discount factor should be directly proportional to the current reliability R_j of the nodes. Together with both the degradation and current reliability metrics, a node computes its discount factor γ as

$$\gamma_j = R_j \frac{2}{\pi} \arctan\left(\frac{\gamma_0}{D_j}\right) \tag{4.19}$$

where γ_0 is a predefined constant. We use inverse tangent function to contain the value of γ_j in the interval [0,1]. Using this discount factor, Q-value is updated based on the current reliability of the nodes, the amount of degradation they will experience due to retransmissions, and networking performance (by the implicit use of ETX in the DEG function). The information is discounted for each node it passes through and is also discounted according to the reliability and the expected degradation of the nodes. In this way, we ensure that the route selected has less hops and is more reliable.

Intuitively, by designing such a discount factor we do not only consider individual node reliabilities at each single hop, on the contrary, we consider the overall system reliability from the source to the destination. From Equation (4.10) and (4.12), we observe that the discount factor is multiplied for each node Q-value is computed. A cumulative equivalent discount factor for the overall system, from a source node *src* to a destination node *des*, can be expressed as $\gamma_{eqv} = \prod_{j=src}^{des} \gamma_j = \prod_{j=src}^{des} R_j \frac{2}{\pi} \arctan(\frac{\gamma_0}{D_j})$. This multiplicative property of the discount factor exhibits a meaningful resemblance to a common system reliability definition used in industry. The system of n components fails if any of its components fails – as we have assumed in our network lifetime definition – for systems organized in a 'series' structure. The series system reliability is then given as follows[55]:

$$R_{system}(t) = \prod_{i=0}^{n} R_i(t) \tag{4.20}$$

at any time t throughout the operation of the system. It can be seen that the cumulative discounted rewards over a packet route has a similar form to Equation (4.20).

In our work, we model the dynamic factors such as ambient temperature and computation workloads of IoT devices, as well as their effect on reliability. We assume that IoT devices run various workloads which contribute to their heating, combined with the thermal stress imposed by the environment. Through this reliability-aware learning parameter, our Q-learning approach is able to adapt these variations in the network and discover better routes without having to know

the network topology and traffic patterns in advance.

4.5 R3-IoT Protocol Design

In this section, we describe several design considerations for our routing protocol, which is a modified version of the Ad-Hoc On-demand Distance Vector (AODV) [151] protocol. We explain implementation details including the route discovery mechanism, the packet structures, the routing table, and the metadata exchanged between devices. We also discuss the overhead involved. Finally, we present results on a small 'toy' example to demonstrate the route discovery process of our approach.

4.5.1 Route Discovery Mechanism

Route discovery is carried out by Route Request (RREQ) and Route Reply (RREP) packets as in the AODV protocol. The source node floods the network with RREQ packets, which are forwarded through multiple hops until they reach to the destination. The RREQ packets are only forwarded to the neighbor with the highest Q-value. When the RREQ packet reaches the destination, the destination node returns an RREP packet through the same path that the RREQ packet followed. In the original AODV protocol, each node broadcasts RREQ packets to all of its neighbors. We choose where the packet should be forwarded independently at each node based on greedy exploitation, that is, the packet is only sent to a single node of highest Q-value. This significantly reduces the number of packets need to be communicated as there is no need for broadcasting.

4.5.2 Packet Structure and Metadata Exhchange

To determine the ETX metric of a link between two nodes, ETX based protocols (e.g., AODV-ETX [171]) use Low Power Payload (LPP) packets. The successful packet delivery probabilities $p_{i\mapsto j}$ and $p_{j\mapsto i}$ in Equation (4.15) are estimated with the LPP packets that are broadcasted over the network. Each node broadcasts LPP at an average period τ and remembers

the LPPs received from its neighbors over the last w seconds, allowing to compute the probability $p_{j\mapsto i}$ at any time t through Equation (4.21). Using relatively small size LPPs, this process incurs only a marginal overhead and saves bandwidth.

$$p_{j\mapsto i} = \frac{count(t, t-w)}{w/\tau} \tag{4.21}$$

where $count(t, t-w)$ is the number of LPPs received by node i and w/τ is the number of LPPs that was sent by node j during the window w. For the link $i \mapsto j$, this allows node i to simply estimate $p_{j\mapsto i}$ by counting successfully received LPPs from j. On the other hand, to compute $p_{i\mapsto j}$, node j includes the number of LPPs it received from i sent during the last w seconds in its each LPP. This way, node i can also estimate $p_{i\mapsto j}$ by using this information.

Table 4.1. Structure of the LPP packet.

Field Name	**Size**	**Description**
Type	8b	Indicates that the packet is of type LPP
LPP ID	8b	Identification number of LPP
Originator IP Addr.	32b	IPv4 address of the node that generates the LPP packet
Originator Seq. No.	32b	Sequence number of the node that generates the LPP packet
No. of Neighbors (n)	8b	The number of neighbors whose LPPs are received by this node
Neighbor IP Addr.	n*40b	IPv4 address of the neighbor from which an LPP packet is received in the last w seconds
Forward LPP Count		The number of LPP packets received in the last w seconds from the neighbor
Max Q-Value	32b	The max Q-value among the neighbors of the node that generates the LPP packet
Reliability	32b	The current reliability of the node that generates the LPP packet

We modify the LPP packets to also include the fields that are needed for Q-learning. The structure of our modified LPP packet is shown in Table 4.1 with the description of the fields. The fields *Neighbor IP Address* and *Forward LPP Count* are repeated for each neighbor. The number of neighbors is indicated by n. *Max Q-Value* and *Reliability* are floating point numbers, so are not suitable for serialization and deserialization of packets. We represent these values as integers

by 10^k for k decimal digit resolution before the serialization and deserialization processes.

4.5.3 Routing Table

In AODV, the routing table entries are classified by the destination addresses. If there are more than one route to the destination, the best route is selected as the one with the least number of hops. We extend the original routing table of AODV with an additional Q-Table. Since we assumed that there exists an agent at each node for completely distributed Q-learning, every node has to store and maintain the Q-values of its next-hop neighbors. We use a dynamic Q-Table, such that the size of the Q-Table of a node is determined by the number of destination nodes and neighbor nodes.

In addition to Q-values, Equations (7)-(12) show that R, ETX, and the maximum Q-Values of each neighbor are needed for action-value updates. Every node has a table for their neighbors with records of this metadata, which is extracted from the LPP packets they receive. Also, the table needs new fields regarding the forward and reverse LPP counts for each neighbor of a node to calculate ETX. Overall, the table has the following fields per each neighbor:

- *NeighborIpAddress:* contains the IP address of the neighbor,
- *ReverseLppCount:* tracks the number of LPP packets received from the neighbor with respective IP address,
- *ForwardLppCount:* tracks the number of LPP packets that the neighbor with respective IP address received from this node,
- *QValue:* contains the Q-Value for the neighbor with respective IP address,
- *MaxQValue:* contains the maximum Q-value in the Q-table of the neighbor with respective IP address,
- *Reliability:* contains the reliability value of the neighbor with respective IP address.

The maintenance of the entries of Q-Tables is ensured through LPP packets. Each node exchanges information with neighboring nodes and updates its Q-Table periodically. *ReverseLpp-*

Count is obtained by counting received LPPs from each neighbor. *ForwardLppCount* is obtained from the received LPP packets. ETX metric is calculated for each neighbor from these two values, as shown in Equation (4.21). The entry, *MaxQValue*, contains the maximum Q-value in the Q-Table of the neighbor. As shown in Table 4.1., the neighbor node finds the maximum Q-value of its own neighbors (i.e., $\max_{k \in N_j} Q_j(des,k)$) and puts it in the LPP packet. This value is received and stored, then is used for learning updates.

4.5.4 Protocol Overhead Analysis

The overhead of communication comes from metadata exchanging with LPP packets. As Table 4.1 shows, the extra header of each LPP packet includes Q-value, maximum Q-value, and reliability. All the other metadata is already present in the default LPP packet structure. All added fields are represented with a 32-bit word. Therefore, in total, there are additional 24 *bytes*. The frequency of broadcasting LPP packets is usually very low compared to data communication, and hence, this part of overhead is negligible. In our experiments, we set the LPP period $\tau = 1s$ so the LPP overhead is only 24 $bytes/sec$ compared to any ETX based protocol.

For the Q-learning algorithm, each node has to carry out computations to update Q-values when an LPP packet exchange occurs. As shown in Equation (6), this computation is a simple multiply-add operation that introduces very modest delay and power consumption, much smaller than that of communications. The computation overhead is also negligible. Additional computations required by our technique involve only two floating-point additions and multiplications per iteration of Q-learning, which is very minor relative to what is required for the rest of the computation and communication. This overhead scales linearly with the number of node's neighbors. However, even though the number of nodes reach thousands, the number of neighbors per node stay in the order of tens [172]. Most other routing protocols have similar communication and computation overhead. Regarding the training overhead of our Q-learning method, we leverage the distributed structure of Q-learning to reduce this cost where the training overhead per individual node is small and scales with the number of node's neighbors.

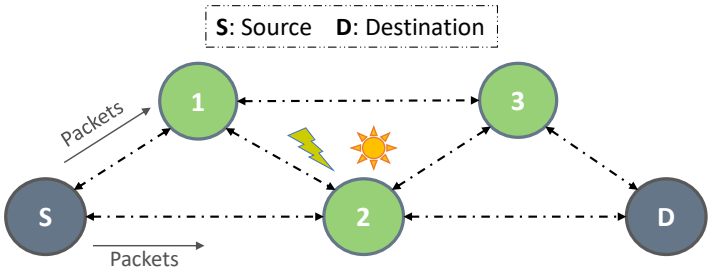


Figure 4.5. Simple network example.

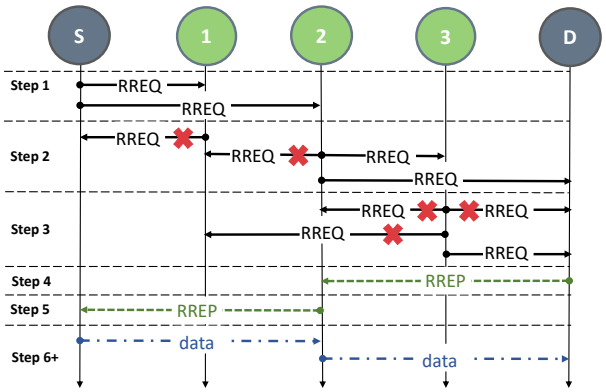
In large networks, the traffic distribution is not expected to be uniform across all nodes in most cases where a fraction of nodes may stay dormant/less active; hence, the frequently accessed nodes become more critical and can be trained faster by levering the distributed nature of our Q-learning. The training of the Q-learning model is done completely online (no offline phase) using temporal difference learning method where the model continues adapting at runtime, using runtime observations.

Besides exchanging metadata and computing Q-values, nodes need to store these metadata – some in the form of routing tables – for all of their neighbors. At runtime, the amount of required metadata storage may vary because the number of neighbors and the communication paths in the network can dynamically change. Again, this metadata is much smaller compared to the storage capacity IoT devices have and the data payload they hold. In the case of very large-scale networks with a high number of nodes, approximate Q-learning can be implemented to dramatically reduce the size of Q-tables. By using function approximation, Q-learning can scale to handle very large state-spaces [173]. In particular, deep Q-learning is a promising solution that proposes neural network function approximation for Q-tables [174, 175].

4.5.5 Small-Scale Example

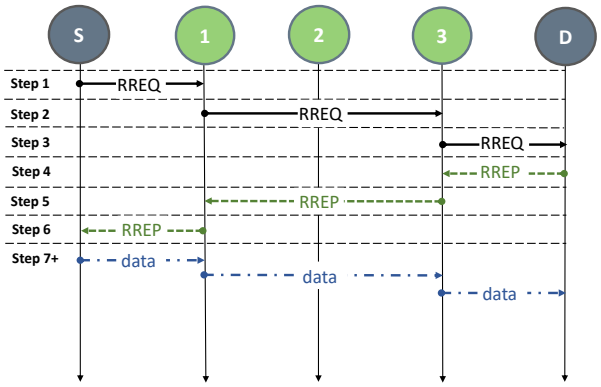
We use the simple network shown in Figure 4.5 to demonstrate the route discovery mechanism of our approach. The source node S wants to send packets to the destination node D. Then, the possible non-cyclic routes that the packet can take are: S-2-D, S-1-2-D, S-1-3-D,

S-2-3-D, and S-1-2-3-D. The particular route to be selected for packet transmission depends on the routing protocol. We assume that at the beginning of route discovery node 2 is highly degraded with a reliability value $R_2 = 0.70$ under the influence of environmental stress, whereas other intermediate nodes have reliability values $R_1 = R_3 = 0.90$. Here, the reliability values depict the probability of not having failures before the given time instant, defined in the interval [0,1]. The route discovery procedure for the default AODV and our proposed approach are shown in Figure 4.6, as a flow diagram of RREQs and RREPs.



AODV Route Discovery Mechanism

(a)



Proposed Route Discovery Mechanism

(b)

Figure 4.6. Route discovery in the simple network example.

In AODV, the source node S first broadcasts an RREQ, which is received by both node 1 and node 2. In our approach instead, RREQs are not broadcasted, they are forwarded to the neighbor with the highest Q-value. As node 2 has the lower reliability in this example, its

Q-value is small as well. Therefore, an RREQ is sent to node 1. As a second step, AODV again broadcasts RREQs from each node that received RREQ in the previous step. If the received RREQ packet has the same ID which was already seen, it is discarded, else, the 'hop count' value of the RREQ is increased by 1, and the packet is broadcasted again until it reaches the destination. Since in our approach RREQs have only a single receiver at each step, they are not discarded at any point. Finally, for both approaches, when the destination node D receives RREQ, it generates a unicast RREP packet and sends it back to the source node. When the source node receives this RREP, it then has the route to the destination and can start sending data packets.

For the example network in Fig. 4.5, the default AODV protocol chooses the least hop path S-2-D, whereas our proposed approach chooses path S-1-3-D, avoiding the most degraded node 2. Results show that node 2 degrades much faster than the other nodes using default AODV and becomes the bottleneck for the network lifetime. When our approach is used, degradation of node 2 slows down and the reliabilities of all nodes meet at the value $R = 0.42$. After that point, all nodes degrade at the same rate. In this way, the MTTF of the simple network is increased by 2.55x. For this particular small network of 5 nodes, this improvement comes with a cost of one extra hop in the path (from one hop to two hop route) and an increase of 16.7% end-to-end delay. In general networks have many more hops from the source to destination than this example. The performance penalty of our approach is not as large with only a few extra hops, for which we provide detailed results in the following section.

4.6 Evaluation

We conduct simulations based on a scenario of environmental monitoring. We refer to an example of real-world deployment, the High-Performance Wireless Research and Education Network (HPWREN) [76]. HPWREN is a heterogeneous wireless sensor network, deployed in the Southern California area. In HPWREN, there are many types of computing systems ranging

from the small wireless sensor nodes, single-board computers, to the high-performance server systems at the UCSD Supercomputer Center. It comprises several subnetworks, but we only focus on a $2km^2$ region of the Santa Margarita Ecological Reserve (SMER) network covered with a mesh topology [176]. We use data collected from HPWREN to model IoT devices in ns-3 and configure the parameters of the simulation. In our simulations, IoT devices (nodes) are randomly distributed over a field of 1000m x 1000m. We conduct experiments for networks of 50, 100, 150, 200, and 250 nodes. A subset of 20 nodes generate Constant Bit Rate (CBR) traffic–typical of sensors that sample at regular intervals–and transmit UDP data with 512 bytes packets to a sink node in an ad-hoc fashion. We chose data rates of 20, 40, 60, 80, 100, and 120 kbps for evaluation. Wireless links between nodes are assumed lossy and have a bandwidth of 2 Mbps, so successful packet transmission is not guaranteed. The lossy communication environment is simulated using the *HybridBuildingsLossModel* in ns-3. Experiments last 730 seconds, are repeated 100 times with different random seeds and averaged to reduce the randomness in results for achieving high confidence. In our experiments we use the IEEE 802.11b standard for the MAC layer because it the most matured communication standard implementation in ns-3. There are efforts on modeling low-rate and low-power standards for IoT, but they are not fully developed yet. Hence, we modify the 802.11 PHY and MAC layer parameters and scale data rate and power values to imitate communication in an IoT environment. All the communication-related parameters used in our simulations are summarized in Table 4.2. For Q-learning, we set the learning rate $\alpha = 0.8$, the discount rate constant $\gamma_0 = 10^{-5}$, and the exploration parameter $\varepsilon = 0.1$. The respective values were determined by carrying out a grid search and finding the best performing values. Therefore, the values used in this chapter are optimized and represent the best achievable results using our method.

Environment: Reliability heavily depends on the temperature of the environment, so we consider realistic, varying ambient temperature conditions. We use a temperature dataset which contains half-hourly ambient temperature measurements of 210 locations over a year [141]. Moreover, we consider the effects of the device being placed in different locations by selecting

Table 4.2. Simulation parameters

Parameter	Value
Simulation Area	1000m x 1000m
Number of Nodes	50, 100, 150, 200, 250
Routing Protocol	AODV, AODV-ETX, R3-IoT
MAC Layer	IEEE 802.11b
Traffic Type	CBR UDP
Data Rate	20, 40, 60, 80, 100, 120 kbps
Packet Size	512 bytes
Bandwidth	2 Mbps
Loss Model	ns3::HybridBuildingsLossModel

the temperature as $T_{amb} \pm U(-10,+10)$, where U is a uniform distribution. For example, a device placed in a closed container under the sun, with airflow around the device is restricted, will have a much higher ambient temperature than a device placed under a shade in open air. Reliability is evaluated considering the TDDB failure mechanism [12], which is a commonly used model in industry today. Other reliability modes can easily be added as needed, since they all exponentially depend on temperature. We scale the impact of reliability degradation in the simulations to reflect 1 year of degradation for 365 seconds of simulation time.

Target Platforms: To capture the heterogeneity of IoT networks, we use models of 3 different embedded devices in our simulations. The target IoT devices are Raspberry Pi 0, Raspberry Pi 2, and ESP8266 microcontroller. We estimate the CPU and WiFi power consumption and temperature of the edge devices by collecting measurements of various applications under different ambient temperatures on the actual devices. We configure the parameters of the heterogeneous node models in ns-3 so that they follow the power, temperature, and reliability characteristics of the modeled platforms. We randomly choose the number of each device platform in the network for every simulation instance.

Workloads: We assume IoT devices run various computation workloads as well as communication data. Therefore, device reliability is also affected by these workloads, which can be considered as an external factor. In our experiments, we consider the ML classification and regression tasks characterized for edge computing settings in [41] with their corresponding power consumption values. The simulated workloads are generated randomly from the measured

set of tasks with the execution times sampled from an exponential distribution with a mean of 10 seconds.

We compare our proposed approach (referred to as R3-IoT) with the following techniques:

- *AODV* [151] is the original Ad-Hoc On-demand Distance Vector routing protocol. It routes the packets through the least hop path.
- *AODV-ETX* [171] finds the path with the least total expected transmission count. The goal is to optimize the packet delivery ratio.
- *Q-AODV* [177] is a Q-learning based energy-aware maximum lifetime routing approach. Network nodes learn to adjust its route-request packets according to their energy profile.

All the models and solutions are implemented in the ns-3 network simulator, leveraging *RelIoT* framework [166] for power, temperature, and reliability simulation. We modified the original models in this framework to support our routing implementation. We simulate the following three different scenarios:

(1) *Constant Uniform Ambient Temperature and No Computation Workload.* When the ambient temperature is the same for all nodes and they do not run any computation, it is expected that only communication would make a difference in their reliability and hence lifetime. In this experiment, we evaluate how good our approach is without the influence of external factors (workloads, ambient temperature) affecting reliability.

(2) *Varying Ambient Temperatures and No Computation Workload.* The most important external factor that affects reliability is the ambient temperature. Since it is an uncontrollable element, the routing decisions would not change it. However, the routing protocol should still be aware of the impact of ambient temperature on reliability over time to be able to moderately utilize the nodes which are under environmental stress. In this experiment, we evaluate if our approach can learn the pattern of ambient temperature changes and the magnitude of its impact on reliability over time.

(3) *Varying Ambient Temperatures and Stochastic Computation Workloads.* Computation is an orthogonal component to communication, which also degrades the reliability of IoT devices. In this case, we model the possible workloads that can be running on IoT devices. These workloads cause extra heating of the device. The routing protocol should learn to avoid the devices which heat up because of running high workloads and operating under high ambient temperatures. We randomly pick workloads from our task dataset and set their service times by sampling from an exponential distribution. Not all nodes do computation, we also randomly select the ones that run workloads.

In the following, we present results for reliability (probability of not having failures up to the given time) and performance (packet delivery ratio, end-to-end delay, and convergence rate). The reliability results are represented as probabilities that take values in $[0, 1]$.

Reliability Results: Fig. (4.7-4.9) show reliability results for different data rates and number of nodes in all scenarios, averaged over randomized simulation runs. The data rate results were collected for networks with 100 nodes, then different numbers of nodes were tested at a data rate of 100 kbps. The reliability values denote the *minimum* reliability in the network since we have defined the network lifetime as the time which the first node fails. In all scenarios and for all approaches, reliability degrades with increasing data rate as expected due to the increased communication activity. AODV performs the worst because it always chooses the same nodes which result in the least hop in the communication path. AODV-ETX changes the nodes used because of randomness in expected transmission counts due to link qualities. This is the reason why it performs similar to R3-IoT in Scenario 1 (Fig. 4.7), because the only source of degradation is due to communication and retransmissions due to link losses. Q-AODV also shows a similar characteristic to AODV-ETX because energy consumption and expected transmission count is highly correlated, so they inherently have parallel goals. Again in Scenario 1, for the increasing number of nodes, there is not much difference in the minimum reliability in the network because the most degraded nodes in this scenario are always the ones closer to the sink.

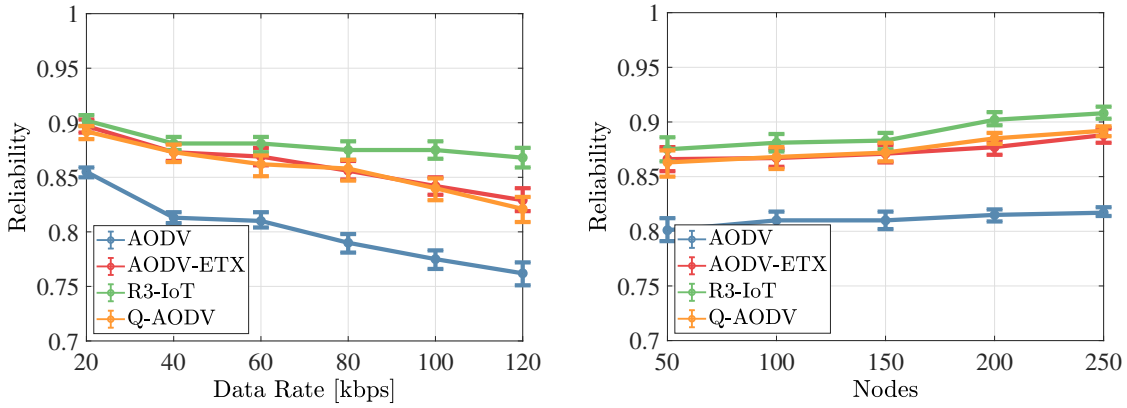


Figure 4.7. Scenario 1: Reliability for different data rates and number of nodes.

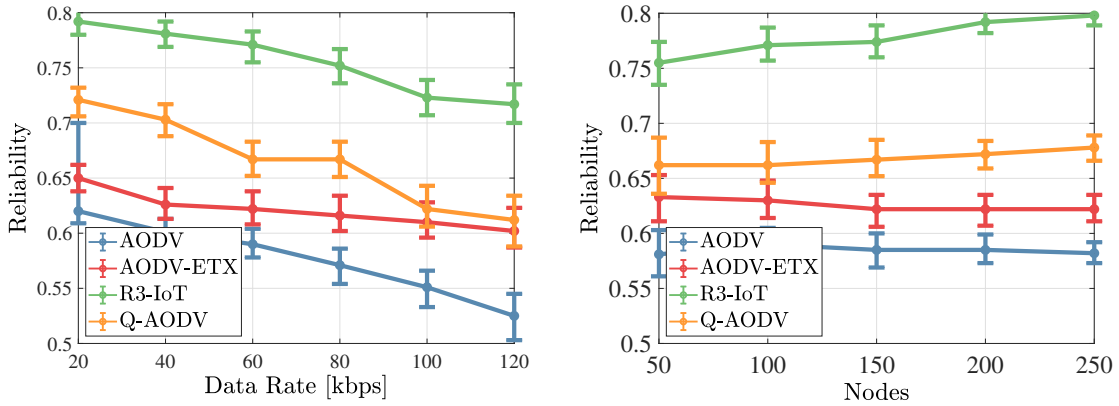


Figure 4.8. Scenario 2: Reliability for different data rates and number of nodes.

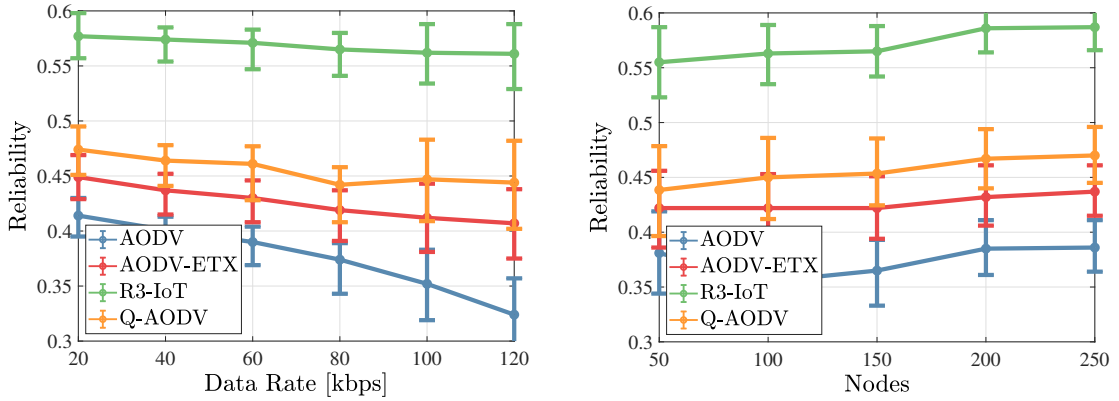


Figure 4.9. Scenario 3: Reliability for different data rates and number of nodes.

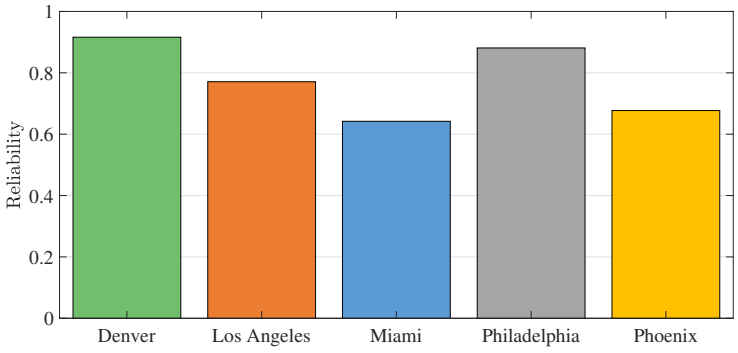
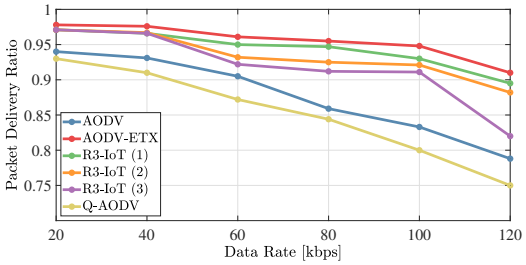


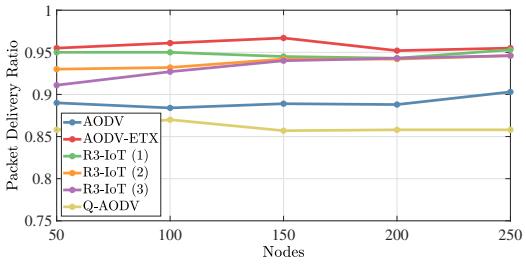
Figure 4.10. Scenario 2: Reliability for cities with different average ambient temperatures.

As shown in Fig. 4.8, when an external factor, temperature, that affects reliability is involved, R3-IoT performs much better than both AODV, AODV-ETX, and Q-AODV. R3-IoT learns the routes that bypass the nodes under high thermal stress. Q-AODV also results in fairly high reliability because the nodes that are highly degraded usually also has lower energy. Hence, Q-AODV avoid the low-reliability nodes too. When there are a high number of nodes in the network, then R3-IoT can find more routes that include only high-reliability nodes. The discrepancy between R3-IoT and other approaches is further exacerbated when computation workloads are added to nodes (Fig. 4.9). In this scenario, at the highest data rate 120 kbps, the most degraded node has 73.2%, 37.8%, and 26.4% more reliability with R3-IoT compared to AODV, AODV-ETX, Q-AODV approaches respectively. Similarly, for the networks with 250 nodes, improvements of 51.8%, 38.6%, and 24.9% are observed.

Finally, for scenario 2, we run simulations under various average ambient temperature values to show its impact on resulting reliability. We use hourly values from 2 years long temperature data collected for cities: Denver, Los Angeles, Miami, Philadelphia, and Phoenix. The respective 2 year average temperatures between 2015-2017 of these cities are: 9.8°C, 17.8°C, 25.1 °C, 12.4 °C, and 22.4°C. Fig. 4.10 shows that a network deployed in a city with hot weather (Miami) can have as much as 42% less reliability after 2 years compared to a network deployed in a city with cold weather (Denver).



(a) Packet delivery ratio vs data rate



(b) Packet delivery ratio vs number of nodes

Figure 4.11. Performance results.

Performance Results: We evaluate the performance of all methods under all scenarios. Fig. 4.11a shows the packet delivery ratios (PDR) for different data transmission rates. Error bars are omitted for performance results since the variation between simulation runs was negligible unlike the reliability results. For all cases, PDR drops with the increasing transmission rate, however, the drop is more severe for AODV and Q-AODV. Both AODV-ETX and R3-IoT chooses better communication links because they utilize the ETX metric, which results in a fewer number of packet losses over links. For all three scenarios, AODV, AODV-ETX, Q-AODV performances do not change because they are agnostic of the changes in node temperature and workloads. Therefore, we use only a single line on the plots to depict the performance of these approaches on the three scenarios. On the other hand, as temperature increases and nodes start running workloads, R3-IoT favors the nodes with higher reliability in the route instead of the nodes with higher quality links. This is the reason performance drops slightly from scenario 1 (R3-IoT (1) on the plot) to scenario 3 (R3-IoT (3) on the plot).

In Fig. 4.11b we compare the PDRs for varying numbers of nodes in the network. Similar to the previous case, Q-AODV performs the worst while R3-IoT performs very close to AODV-ETX. There is a slight increase in PDR as the number of nodes increases for all approaches. This is due to the fact that there are more possible routes that packets can take if the number of nodes is high. For more nodes, the performance of R3-IoT in both Scenario 1 and Scenario 2 approaches to its performance in Scenario 1. Even though R3-IoT tends to choose the nodes with

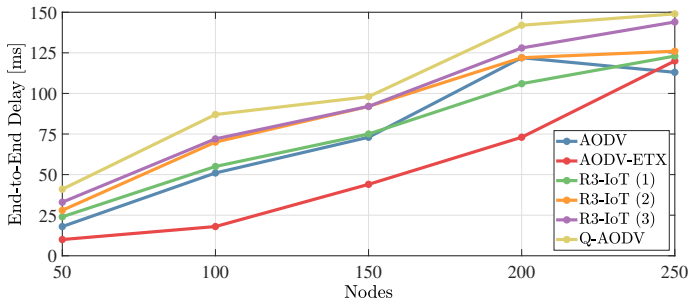


Figure 4.12. Average end-to-end delay.

high reliability instead of the ones that can result in better performance, there are more nodes that satisfy both properties in a network with large number of nodes.

We further present comparison for average end-to-end delay of all methods under all scenarios in Figure 4.12. The results are plotted only for varying number of nodes because no significant changes were observed for different data rates. Similar to PDR results, AODV, AODV-ETX, Q-AODV delays do not change under varying node temperature and workloads. Q-AODV performs the worst because it does not include any mechanism to determine the shortest paths or the most reliable links. AODV and AODV-ETX can find paths with minimal number of hops and hence acquire very small delays. Our approach is similar to AODV in Scenario 1, but it gravitates toward balancing reliability when ambient temperature and workloads are introduces to the simulations.

Fig. 4.13 shows the convergence performance of our Q-learning approach for networks with varying number of nodes. The corresponding experiments were conducted at a data rate of 100kbps. We omit the results for varying data rates as we have not observed any significant changes at different data rates, the convergence rates were similar across the board. The values plotted are the normalized sum of the Q-values of all the nodes in the network. Epochs denote the number of Q-value updates, that is, the operation described in using Equation (4.12). The time each epoch takes can be configurable in our approach and is equal to the LPP packet period since Q-value updates take place when LPP packets are exchanged. In the second plot we

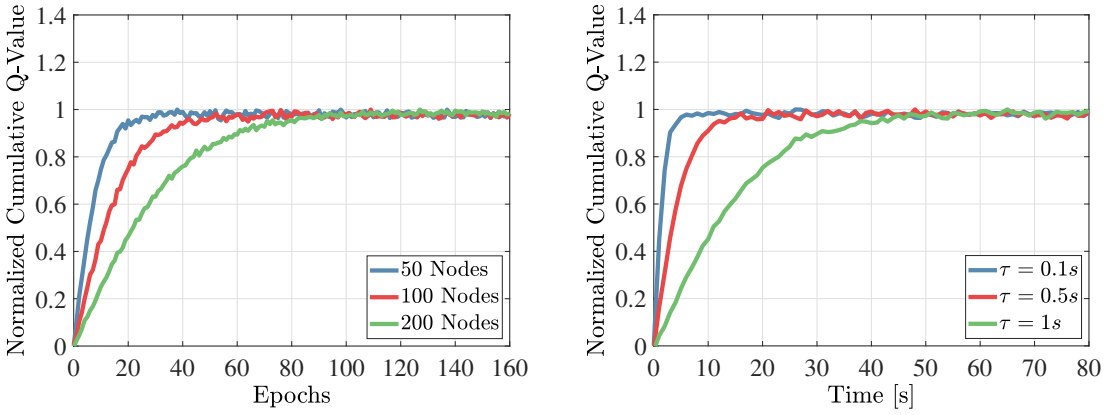


Figure 4.13. Convergence of Q-values

present convergence in actual clock time metric for different LPP periods, simulated on a 100 node network. Results show that convergence is reached in fairly low iterations, especially for networks with a small number of nodes. This implies an efficient use of samples, that is, the sampling complexity of the proposed distributed reinforcement learning approach is low. It should be noted that these results show a learning curve for a "cold-start", when the algorithm is run right after the network is set up. Similar initialization periods are common for many routing protocols for building their routing tables. After the initial phase, it only needs a few epochs for the Q-learning algorithm to react to the changes in the system (e.g. variations in the external factors such as ambient temperature and workloads).

4.7 Conclusion

In this chapter, we explored the problem of maintaining IoT device reliability from the perspective of network routing. The literature on network routing up to date did not study hardware related reliability issues. We proposed a distributed reinforcement learning based routing approach to improve network MTTF, which learns to make its decisions based on the current reliability of the nodes, the amount of degradation they will experience, and networking performance. We extended the ns-3 AODV protocol with a Q-learning algorithm and demon-

strated improved network MTTF compared to AODV, AODV-ETX, and Q-AODV methods. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance.

Chapter 4 contains material from “Reliability-Aware Routing in IoT Networks Using Reinforcement Learning”, by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in Ad Hoc Networks [3]. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Dynamic Optimization of Battery Heath in IoT Networks

The reliability and maintainability of the Internet of Things (IoT) devices become highly important as the number of "things" grows rapidly. The majority of the IoT devices have batteries which age, degrade, and eventually require maintenance. Existing work focuses on ensuring that batteries have sufficient amount of stored charge to operate until they can recharge, but does not consider battery degradation. This leads to high replacement and maintenance costs in large IoT networks. In this chapter, we formulate the problem of minimizing battery degradation to improve the lifetime of IoT networks and solve it with Model Predictive Control (MPC) leveraging models for battery dynamics and State of Health (SoH). The battery SoH is modeled using a realistic non-linear model while taking ambient temperature into account. We demonstrate that our solution can improve network lifetime up to 68.5% compared to conventional energy consumption focused algorithms, which use simple linear battery models. The proposed approach achieves near-optimal performance in terms of preserving battery health, staying within 8.7% SoH with respect to an ideal oracle solution on average.

5.1 Introduction

The Internet of Things (IoT) is a growing network of heterogeneous devices that have the ability to process and transfer data. IoT will connect more than 20 billion "things" by 2020

according to Gartner Inc.[178]. When IoT is fully realized, the maintenance and diagnostics costs will be enormous, and if not addressed, it can limit the scalability of IoT solutions [179]. Since the majority of these devices are battery-powered, a part of these costs is associated with battery maintainability. Even though battery itself might be cheap, battery replacement, especially for large-scale IoT systems, is often not feasible due to logistical constraints. One example is the High-Performance Wireless Research and Education Network (HPWREN), which have battery-powered sensors deployed on canyon walls and mountain peaks with no road access [76]. In such cases battery replacement involves expensive labor & infrastructure, hence the battery life should be improved to keep the network running for as long as possible.

There is often confusion when discussing battery lifetime because the lifetime for rechargeable and non-rechargeable batteries are described in different ways. Non-rechargeable batteries die, and need to be replaced after their initial charge is completely depleted. Therefore, the indicator for remaining battery life is the State of Charge (SoC). On the other hand, rechargeable batteries can withstand hundreds of charge-recharge cycles, allowing operation for extended periods when combined with energy harvesting solutions, such as solar cells or thermal energy. Despite their ability to be recharged, these batteries still have limited lifetimes, and require replacement due to aging. In this case, instead of SoC, we need to consider their State of Health (SoH) which is a figure of merit of the physical condition of a battery. SoH degrades due to cycle aging (charge-discharge rate & total amount) and calendar aging (ambient conditions, e.g. temperature) which results in deterioration of battery conditions in the form of internal impedance increase, open voltage decrease, and most importantly, capacity fading. Depending on its application, a rechargeable battery reaches its end of life with an SoH between 70%-80% and needs to be replaced.

Most works in the literature concerning lifetime maximization either consider non-rechargeable batteries and deal with SoC, or assume ideal operation for rechargeable batteries, neglecting the effects of SoH degradation in their management strategies. Our main insight is that if maximum lifetime is targeted in the network, specifically the battery SoH should be

considered. Particularly, we observe that the techniques which focus on optimizing the energy consumption of a network do not yield optimal battery life. In this work, we formulate the problem of determining the data flow that minimizes SoH degradation of rechargeable batteries for an IoT network where battery-powered devices have the capability of sensing, processing, and communicating data. The amount of data routed through a device affects the power consumed for communication & computation, which in turn influences the rate of degradation. In our formulation, we model batteries from two different perspectives:

- **Battery Degradation:** The focal point of this work is the fact that battery SoH degrades at different rates depending on how the battery is used. In the light of this, we can intelligently manage the network to prolong battery lifetime. Hence, we have a model that relates current rate, SoC, depth of charge/discharge and temperature to how SoH degrades.
- **Battery Dynamics:** We use the *Temperature Dependent Kinetic Battery Model* (T-KiBaM) [108], a dynamic model which can describe the nonlinear characteristics of available battery capacity. 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 [107]. Therefore, to realistically capture the influence of power consumption on the battery, it is inaccurate to assume linear energy depletion with respect to the power consumed/current drawn, and a dynamic battery model is needed.

As a result of this dynamic behavior, the solution to our problem considers the battery state over time and therefore, is time-dependent rather than fixed. Hence, we adopt an optimal control formulation and propose a model predictive controller (MPC) solution to dynamically control data flow rates in the network to minimize SoH degradation over a predefined horizon. We evaluate our solution using real-world deployment in a smart home and a large scale IoT network HPWREN. We show that our solution can achieve comparable performance to an "oracle" solution which knows all future data. For comparison, we implement a standard network

lifetime maximization method [62] which adopts an ideal battery model with linear energy depletion. We also investigate the impact of ambient temperature on SoH degradation and network lifetime. Furthermore, an example extension to the original problem is presented by regularizing the objective function with an end-to-end delay function.

The rest of the chapter is organized as follows. In Section 5.2, we review related work on maximum network lifetime routing and battery degradation management. In Section 5.3, we first start with outlining the overall problem and describing our network model. Next, we build the battery dynamics and investigate the mechanisms behind battery degradation to obtain a closed-form, nonlinear mathematical expression for the SoH of a battery. Lastly, we construct a finite horizon optimal control problem with the goal of determining the data flow to minimize the degradation of an IoT network and present our MPC solution. In Section 5.4, we provide experimental results and conclude our work by discussing these results.

5.2 Related Work

There is a significant amount of literature addressing the lifetime of Wireless Sensor Networks (WSNs) and IoT networks. Publications in that area usually consider non-rechargeable batteries with limited energy and maximize the time at which the batteries drain out of energy [62],[63]. A common issue with such techniques is that they do not consider the battery dynamics and find a static route based on linear battery energy depletion assumption. Recent studies [64],[65] involved battery dynamics that are able to capture the "non-ideal" behavior of actual batteries in their optimization formulations. Even though they show that one can achieve a significantly longer lifetime with an optimal routing policy using a non-ideal battery model, the solution does not suit systems with rechargeable batteries.

Another set of publications investigate energy harvesting networks with rechargeable batteries. This work usually tries to develop control algorithms to optimally utilize available energy [66],[67]. However, only a handful of studies consider the degradation of batteries, which

is the major factor in determining the lifetime of a network of devices with rechargeable batteries. A Markov model based mathematical characterization of harvesting-based battery-powered sensor devices was provided in [68], particularly focusing on the impact of battery discharge policy on degradation. The authors show that by using this model, a degradation-aware policy significantly improves the lifetime of the sensor compared to "greedy" policies. We instead search for network-level controls (i.e. routing) compared to finding a policy for single sensor node/device. In [69], the issue of battery degradation is approached from a MAC protocol design perspective. Random MAC protocols can generate bursts of transmissions and idleness which may increase battery degradation rate. To solve this problem, they propose an aging aware binary exponential backoff algorithm to avoid excessive fluctuations. This study is tangential to our work since it touches upon the degradation problem with a small modification on MAC protocols. More recently, a technique was presented in [70] to predict SoH in WSN applications from various battery related parameters, which can contribute to building degradation-aware management strategies for IoT networks.

The degradation of batteries in a network control problem is studied primarily in battery energy storage systems, smart grid, and data centers. In [72], the authors include the battery degradation processes in the optimization and propose a linear programming approach for optimization of degradation & performance in offgrid power systems with solar energy integration. In [71], a model predictive control (MPC) based algorithm with an explicit cost function considering battery degradation is implemented for battery energy storage systems. A recent paper [73] presents a distributed control method that can handle multiple batteries connected to the grid using a high accuracy nonlinear battery model. In the context of data centers, [74] and [75] use nonlinear Lithium-ion battery health degradation model for health-aware optimal control. However, these work are not directly applicable to IoT domain because of the different structure of the network, and additional constraints that the network possesses. In those areas, batteries are often modeled in aggregate fashion. In IoT networks, the batteries from different devices are not physically connected and can only supply energy to the associated device. The devices

work together to accomplish a network-level task, but their energy demands are individual which differentiates other domains from IoT.

Network lifetime studies up to this point have been mostly State of Charge (SoC) optimization for non-rechargeable batteries with ideal linear models for battery dynamics. The ones that study rechargeable batteries focus on energy management strategies to optimally utilize energy harvesting solutions. Along with just a few other works, we investigate the State of Health (SoH) of batteries. Complimentary to previous works in this area, we control the network to optimize its lifetime by minimizing SoH degradation. To perform a more accurate optimization, we incorporate battery models which capture the temperature-dependent, nonlinear charging/discharging and degradation behavior into the system model.

5.3 Optimal Nonlinear Battery Control

5.3.1 Problem Overview

The goal of this work is to optimize battery health in IoT networks by controlling data flow rates since each device in the network consumes energy for communication and computation as a function of data flow. The energy amount delivered by the battery depends on both short-term battery dynamics and long-term battery wear. Therefore, while the battery dynamics determine the State of Charge, our control algorithm focuses on optimizing State of Health degradation to ensure long term operation. In the following sections, we start by describing our network model, then build the battery dynamics and investigate the mechanisms behind battery degradation to obtain a closed-form, nonlinear mathematical expression for the SoH of a battery. Table 5.1 provides the list of symbols that are used throughout this chapter.

5.3.2 Network Model

We model the IoT networks with three layers: top, middle, and bottom. The top layer represents the wireless mesh backbone of the network. The bottom layer contains sensor nodes

Table 5.1. Nomenclature

Symbol	Definition
S_i	Set of nodes which node i can send data
$d_{i,j}$	Distance between nodes i and j
$w_{i,j}(t)$	Data flow rate from node i to j
$G_i(t)$	Data generation rate of node i
C_r, C_e, C_c	Reception, sensing, computation energy constants
C_f, C_s	Transmission energy constants
$u_i(t)$	Discharge current of battery i
$r_i(t)$	Charge current of battery i
$i_i(t)$	Net current of battery i
$q_A(t)$	Available charge
$q_B(t)$	Bounded charge
$h_A(t)$	Available charge well height
$h_B(t)$	Bounded charge well height
k	Conductance parameter
C_R	Battery rated capacity
$\delta_i(t)$	Difference between heights of two wells of battery i
$\gamma_i(t)$	Total charge of battery i
$SoC_i(t)$	State of Charge of battery i
V_i	Voltage of battery i
T	Temperature
DoD	Depth of discharge
T_{amb}	Ambient temperature
$SoH_i(t)$	State of Health of battery i
$Deg_i(t)$	SoH degradation of battery i

and the middle layer is composed of a wireless network of gateways. Each gateway node gathers the data coming from the underlying sensors and delivers it to the backbone layer. These nodes can also perform data analysis and processing.

We consider a model with multiple source and gateway nodes, one base station, and fixed topology. The network consists of N nodes, where nodes from 1 to N-1 denote source and gateway nodes and N denote the base station. We assume that the energy supply of the base station is not constrained but all other nodes have a rechargeable battery that can store a limited amount of energy. $SoH_i(t)$ and $SoC_i(t)$ are respectively the State of Health and State of Charge of the battery of node i, $i = 1,...,N$ at time t, and the dynamics of $SoH_i(t)$ is described

with details in the next section. The distance between the nodes i and j are denoted by $d_{i,j}$, and is time-independent since we assume fixed topology. Note that relatively infrequent topology changes can be accounted for by periodically recalculating a new control policy.

Let S_i denote the set of nodes to which node i can send packets. Conditions on S_i can be enforced to constrain the behavior of the network, but we only restrict the transmission distance in our problem. Then, $S_i = \{j : d_{i,j} < d_{max}\}$, where d_{max} is the distance of transmission with maximum power. The notation $j \in S_i$ will be used to show node i can communicate with node j. Let $w_{i,j}(t)$ be the data flow rate from node i to node j at time t. The vector $\mathbf{w}(t) = [w_{1,2}(t), ..., w_{1,N}(t), ..., w_{N,N}(t)]^\intercal$ defines the control vector in our problem. Let $G_i(t)$ denote the information generation rate at node i, then we can express the total information that needs to be communicated to the gateway as $G_N(t) = \sum_{i<N} G_i(t)$.

We assume every node in the network has a sensor, CPUs, digital signal processors and a radio link. Since we are dealing with nodes that are sensing, computing and receiving/transmitting, the key energy parameters that contribute to discharge current $u(t)$ of node's battery are: the energy needed to sense a bit E_{sense}, receive a bit E_{rx}, transmit a bit E_{tx}, and compute a bit E_{comp}. For a given distance $d_{i,j}$ between nodes i and j, we compute the energy expenditure as follows:

$$E_{tx} = p(d), \;\; E_{rx} = C_r, \;\; E_{sense} = C_e, \;\; E_{comp} = C_c \tag{1}$$

where C_r, C_e, C_c are given constants dependent on the communication, sensing, and computation characteristics of nodes respectively, and $p(d) \geq 0$ is a function monotonically increasing in d; the most common such function is $p(d) = C_f + C_s d^\beta$ where C_f,C_s are given constants and β is a constant dependent on the medium [180]. For each sensor node i in the network, we can write

the discharge current $u_i(t)$ as in (2), where V_i denotes the voltage of the battery .

$$u_i(t) = \frac{1}{V_i}\sum_{j\in S_i} w_{i,j}(t)(p(d_{i,j}(t)) + C_c) + \frac{1}{V_i}\sum_{j|i\in S_j} w_{j,i}(t)C_r + C_e G_i(t), \tag{2}$$

5.3.3 Battery Model

Battery Dynamics

In this work we use Temperature Dependent Kinetic Battery Model (T-KiBaM), an extension to KiBaM [108]. T-KiBaM is able to accurately characterize the two important effects (Rate Capacity effect, and Recovery effect) that make battery performance nonlinear [107]. 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 [65] that using battery models which captures these effects results in more accurate optimization algorithms, and leads to improvements in network lifetime.

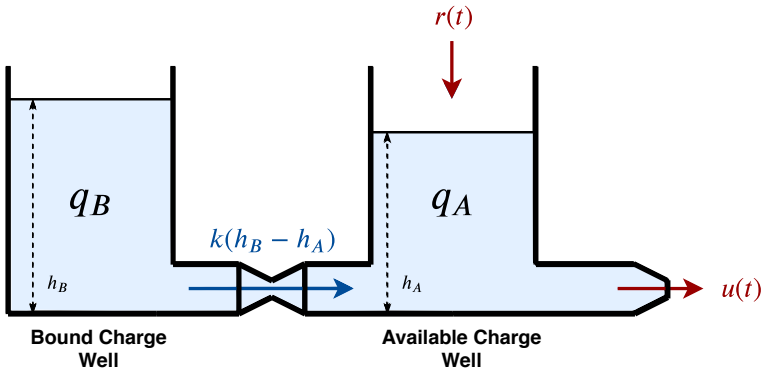


Figure 5.1. Temperature Dependent Kinetic Battery Model

As shown in Fig. 5.1, T-KiBaM models the batteries with two wells, respectively the Bound Charge Well (BCW) and the Available Charge Well (ACW). Three constants are needed

for the model: C_R, the rated capacity of the battery; c, the fraction of capacity that may hold available charge; and k, the rate constant. Initially, a part $q_A(0) = cC_R$ of charge is put in the ACW, and a part $q_B(0) = (1-c)C_R$ in the BCW. The charge flow from BCW to ACW through a "valve" with a conductance $k = k_{Arrhenius} = Ae^{-\frac{Ea}{RT_{amb}}}$, a temperature dependent rate constant given by Arrhenius Equation. A is the pre-exponential factor (in s^{-1}), E_a is the activation energy (in KJ/mol), R is the universal gas constant ($8.314 \times 10^{-3} KJ/mol \cdot K$) and T_{amb} is the ambient temperature (in $Kelvin$). The charge flows creating a current $i(t)$ as long as there is a difference between the heights of two wells, i.e. $\delta = h_B - h_A \neq 0$. The heights of these two wells are given by $h_A = q_A/c$ and $h_B = q_B/(1-c)$. Net current $i(t)$ is the difference between an output $u(t)$ representing the discharge outflow due to workload, and a recharge inflow $r(t)$ such that $i(t) = u(t) - r(t)$. The following system of differential equations describes KiBaM.

$$\begin{cases} \frac{dq_A}{dt} = -i(t) + k(h_B - h_A) \\ \frac{dq_B}{dt} = -k(h_B - h_A), \end{cases} \tag{3}$$

For this work we found it convenient to apply a coordinate transformation to variables for using them in the problem formulation. We transform the variables from q_A and q_B to $\delta = h_B - h_A$ (height difference between wells) and $\gamma = q_A + q_B$ (total charge in the battery). Under this transformation, we can write the new differential equations as:

$$\begin{cases} \frac{d\delta}{dt} = \frac{i(t)}{c} - k'\delta \\ \frac{d\gamma}{dt} = -i(t), \end{cases} \tag{4}$$

where $k' = k/c(1-c)$, with initial conditions $\delta(0) = 0$ and $\gamma(0) = C_R$. In the new coordinate system the condition for the battery to be empty is: $\gamma(t) = (1-c)\delta(t)$, meaning that there is no charge left in the available charge well. The two equations in (5.4) constitute the battery dynamics that is used in optimization problem formulation.

Since batteries provide higher effective capacities at higher temperatures [108], we use a

Correction Factor CF to adjust initial battery capacities (C_R) according to the ambient temperature. CF indicates multiplicative gain or loss of the battery capacity at different temperatures. All parameters (CF, c, k, C_R) can be obtained using the battery data-sheets, and through experimental measurements. In this work, we use the parameters obtained by experimental measurements obtained in [108] for Li-Ion batteries.

SoH Degradation Model

The State of Health (SoH) refers to the condition of the battery and the value of SoH declines from 1 (healthy battery) to 0 (dead battery) over time due to degradation. For the SoH degradation, we employ the model from [181]. Over continuous battery charge/discharge cycles, significant factors that influence the SoH degradation of a battery are temperature T, open circuit voltage V_{OC}, and depth of discharge DoD. Knowing that there is a mapping of V_{OC} from State of Charge (SoC), we can consider three aspects of the battery for estimating its degradation in this model: T, DoD, and SoC. SoC is defined as the portion of available battery capacity at a given time and DoD is used to describe how deeply a battery is discharged. The formulation of SoC directly comes from our Kinetic Battery Model, where we defined γ as the total charge in the battery. The only difference is that SoC represents the normalized charge level of the battery, i.e. $SoC = \frac{\gamma}{C_R} \in [0,1]$. The degradation model used in this work makes two assumptions: 1) Each of these effects is independent of the others, and 2) The effects themselves are independent of battery age.

Since the effects of T, DoD and SoC on degradation are assumed to be independent, we can write the total battery degradation as $Deg_{total} = Deg_{SoC} + Deg_{DoD} + Deg_T$. The SoH at a given time t is $SoH(t) = SoH(0) - Deg_{total}(t)$, and can be expressed explicitly as shown in (**??**).

$$\begin{aligned} SoH(t) = 1 &- [\phi_1 SoC_{avg}(t) + \phi_2] + [\theta_1 (\Delta SoC(t))^{\theta_2}] \\ &- \left[\int_{t'=0}^{t} \sigma_1 e^{-\sigma_2 (T_{amb} + \sigma_3 |i(t')|)^{-1}} dt' + \sigma_4 T_{amb} \right], \end{aligned} \tag{5}$$

The first bracket expression is the term representing the capacity fade degradation attributable to *SoC*. This is based on an approximation that a time period during which the *SoC* with an average of SoC_{avg} has the same effect on battery life as simply staying at SoC_{avg} for the same time period [181]. The second bracket expression is DoD related degradation which accounts for capacity fade resulting from SoC swing, i.e. maximum SoC minus the minimum over an interval. Finally, the last bracket expression is the degradation due to temperature described with a similar exponential model to Arrhenius relation. The temperature change in the battery is given as a linear function of charge current and ambient temperature (T_{amb}). The absolute value of the current $i(t)$ is used so that the expression is both valid for charging and discharging.

We verified our battery degradation model against NREL Li-Ion battery aging dataset [182]. The repository contains various experiment scenarios and physical measurements of cycling batteries until their capacity is reduced below the industry standard of 80% of their original capacity, which is the point where batteries are considered "dead". The coefficients in the expression (5) are obtained by fitting the to the experiments under different temperatures and charge/discharge profiles. We selected $\phi_1 = -10^{-3}$, $\phi_2 = 10^{-8}$, $\theta_1 = 25$, $\theta_2 = 0.017$, $\sigma_1 = 1.4x10^{-4}$, $\sigma_2 = -75$, $\sigma_3 = 0.1$, $\sigma_4 = 4x10^{-5}$. Table 5.2 shows the error compared to the measurements from batteries tested at 3 different temperatures.

Table 5.2. Battery Model Validation

Battery	Temperature	Error
Li-Ion_{25}	4°C	3.1%
Li-Ion_{5}	24°C	1.6%
Li-Ion_{49}	43°C	4.4%

5.3.4 Optimal Control Problem Formulation

Our objective is to minimize the SoH degradation of a network by controlling data flow rates $w_{i,j}(t)$. As a common definition, a network is considered dead when any of the nodes

die. To prevent this, we particularly try to minimize the accumulated degradation on the most degraded node, since it is the one that will fail the first. Hence, the cost function is $\underset{\mathbf{w}(t)}{minimize} \max_{i \in N} Deg_i(T)$, where degradation at end of interval t is decribed by: $Deg_i(t) = SoH_i(0) - SoH_i(t)$.

Next, we define constraints to represent the battery's physical nature:

- Current Limit: The discharge and charge power of a battery is limited, thus there are bounds on charge/recharge current, $Lp_i \leq u_i(t) \leq Up_i$ and $Lp_i \leq r_i(t) \leq Up_i$.

- Charge Limit : The charge cannot exceed the maximum capacity of the battery, and as stated in Section IIIC the condition for the battery to be empty is: $\gamma(t) = (1-c)\delta(t)$. Therefore, the corresponding constraint equation is given as $Lc_i \leq \gamma_i(t) \leq Uc_i$ where $Lc_i = (1-c)\delta(t)$.

Using the battery model, the network model, and constraint equations, the discrete-time optimization problem for a finite interval T is formulated in (6):

$$\underset{\mathbf{w}_t}{\text{minimize}} \quad \max_{i \in N} Deg_{i,T} \tag{6}$$

$$\text{subject to} \quad \delta_{i,t+1} = \delta_{i,t} + \frac{u_{i,t} - r_{i,t}}{c} - k\delta_{i,t},\ \delta_{i,0} = 0, \tag{7}$$

$$\gamma_{i,t+1} = \gamma_{i,t} + (r_{i,t} - u_{i,t}),\ \gamma_{i,0} - C = 0, \tag{8}$$

$$u_{i,t} = \frac{1}{V_i} \sum_{j \in S_i} w_{i,j}(t)(p(d_{i,j}(t)) + C_c) + \frac{1}{V_i} \sum_{j|i \in S_j} w_{j,i}(t) C_r + C_e G_i(t), \tag{9}$$

$$G_{i,t} = \sum_{j \in S_i} w_{i,j,t} - \sum_{j|i \in S_j} w_{j,i,t}, \tag{10}$$

$$0 \leq w_{i,j,t} \leq W_{max}, \tag{11}$$

$$Lp_i \leq u_{i,t} \leq Up_i, \quad Lp_i \leq r_{i,t} \leq Up_i, \tag{12}$$

$$Lc_i \leq \gamma_{i,t} \leq Uc_i. \tag{13}$$

where (7) and (8) are battery dynamic equations with state variables $\delta_{i,t}$ and $\gamma_{i,t}$ representing node i's charge level at the time instant t. Workload in terms of data flow for each node i is expressed by equation (9). Constraints on the control variable $\mathbf{w}_t$ are specified in (10), (11). Finally, (12) and (13) specifies the constraints due to physical limitations of batteries.

Since the solution is based on a finite horizon, two methods are applicable: i) the algorithm is executed once for the complete horizon to get the optimal solution and ii) model predictive control (MPC), where an optimization algorithm is executed at each time interval based on the predicted horizon values and dynamically updated at the next decision interval. Even though the first method gives us the optimal solution for the interval T, it requires knowledge of future (e.g. data generation $G_{i,t}$, current generation $r_{i,t}$, and temperature T_{amb}), thus it is not applicable in

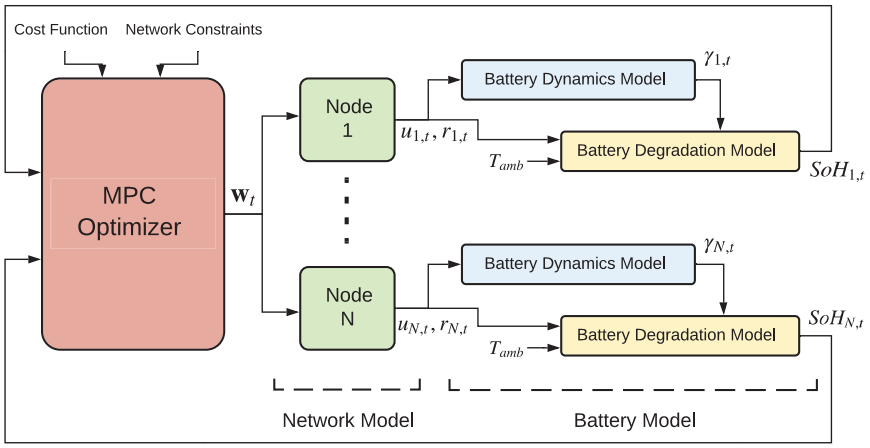


Figure 5.2. Block diagram of the proposed MPC solution

practice. For this reason, we employ MPC with the goal of minimizing degradation across the network, and use the first method as a performance benchmark to compare our solution.

As depicted in Fig. 5.2, the main elements of the discrete time model predictive control are the optimizer and the model. MPC determines the model outputs for the prediction horizon, denoted with M. In the same horizon, the optimizer aims to find the optimal control sequence $\{\mathbf{w}_{k-1+t}, t = 1,...,M\}$ for the cost function (6), subject to problem constraints. Only the first element $\mathbf{w}_k$ of the optimized control sequence is applied to the model and the optimization process is repeated at each time step. It is assumed that we have predictions of energy generation, data generation, and ambient temperature for some time into the future within the horizon of the predictive controller. In other words, given a prediction horizon M, we assume knowledge of $r_{i,t}$, $G_{i,t}$, and $T_{amb_{i,t}}$ for all $t \in \{k,...,k+M-1\}$. When the prediction horizon is less than 24 hours, such an assumption is reasonable as energy generation (e.g solar energy) tends to follow daily patterns and one-day ahead weather predictions can be fairly accurate for ambient temperature.

5.3.5 Regularizations

We can regularize the cost function to obtain many different extensions to the original problem of minimizing degradation. Consider an utility function $\phi(\mathbf{w})$, which can be used to

model power consumption, delay etc. With a trade-off in network lifetime optimality, the cost function can be regularized as follows to improve network performance in other aspects.

$$\underset{\mathbf{w}_t}{\text{minimize}} \qquad \max_{i \in N} Deg_{i,T} + \lambda \phi(\mathbf{w}) \tag{14}$$

In the following, we show the use of a regularization function for end-to-end delay of a network, although many different utility functions are possible.

End-to-end delay: The end-to-end packet delay from a source node to a sink node depends on the number of hops along its path. As the number of hops increase, the packet will be received by the sink with a higher delay. Queuing delay on the nodes can be assumed negligible because the data rate is low enough for most of the IoT applications to make the number of hops the dominant factor. Instead of directly using the number of hops, we create a metric to provide similar behavior. For a given node i, if its neighbor $j \in S_i$ is farther from the sink than another neighbor $k \in S_i$, then the delay for following a path through node j should be greater compared to node k. Thus, we define:

$$h_{i,j} = \frac{d_{j,N}}{d_{i,N}}, \quad i \in 1,...,N-1, \ j \in S_i \tag{15}$$

To attain the lowest delay, a packet must be forwarded to the neighbor with the minimum h value; the one closest to the sink (node N). A delay function for a node i can be given as:

$$\phi_i(\mathbf{w}) = \sum_{j \in S_i} h_{i,j} w_{i,j}, \quad i \in 1,...,N-1, \tag{16}$$

The regularization function should ensure that most of the data traffic is routed through the minimum hop path. Since we are interested in the average delay of the network, the delay function in (16) is summed up over all nodes and averaged over time. The regularization function for end-to-end delay is:

$$\phi(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i\in N-1}\sum_{j\in S_i} h_{i,j}w_{i,j} \tag{17}$$

5.4 Evaluation

5.4.1 Experimental Setup

To illustrate the results of our solution, we consider two examples of real-world deployments: High-Performance Wireless Research and Education Network (HPWREN) [76], and a study on IoT Smart Home developed in our lab. We cover the frequently used mesh and clustered mesh (hierarchical) network topologies with HPWREN and Smart Home cases respectively.

HPWREN

HPWREN is a heterogeneous wireless sensor network, deployed in the Southern California area. In HPWREN, there are many types of computing systems ranging from the small wireless sensor nodes, single-board computers, to the high-performance server systems at the UCSD Supercomputer Center. It comprises several subnetworks, but we only simulate the Santa Margarita Ecological Reserve (SMER) network which covers a region of 2500m x 1250m with a mesh topology. There are a total of 15 cameras and 1 acoustic sensor deployed, each generating data of different sizes and at different sampling rates. Data sizes range from 20kB to 2MB with a sampling interval 30sec to 1hour. The devices are equipped with solar panels that supply energy to recharge batteries. We use real temperature data collected from Vaisala WXT520 weather sensors in our battery models as the ambient temperature (T_{amb}) and real solar radiation data from Davis solar sensors to determine the of amount solar power generation ($r_{i,t}$). Fig. 5.3 depicts the temperature and solar generation profiles of 16 nodes in HPWREN during a day. We estimated the power output of solar panels from solar radiation and used it as the battery charging value for our calculations .

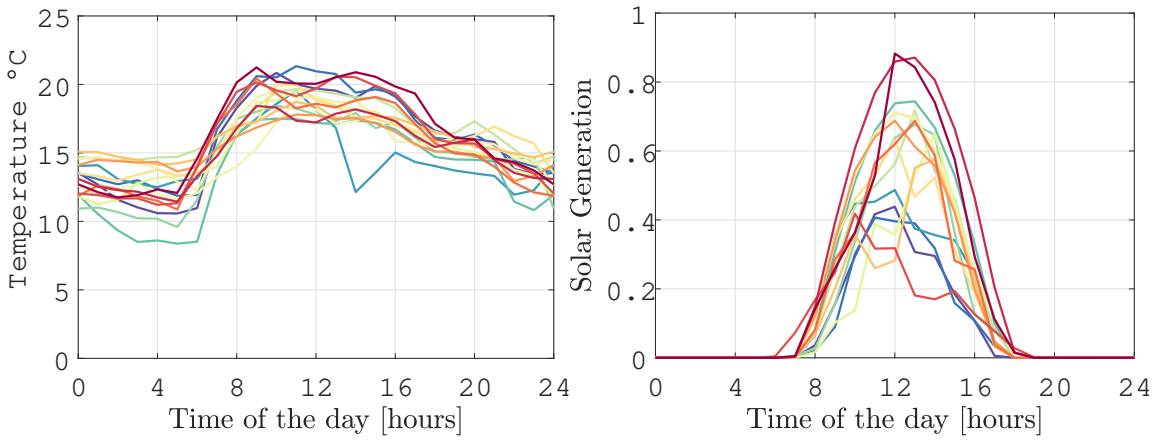


Figure 5.3. Temperature and normalized solar generation of 16 nodes in HPWREN during a day.

Smart Home

We have a house instrumented with several off-the-shelf heterogeneous sensors as shown in Fig. 5.4. Each room in the house has several sensors which help in identifying activities local to that room. These sensors are: (1) kitchen door contact, (2) fridge door contact, (3) kitchen drawer contacts 1&2, (4) teapot smart-plug, (5) kitchen smart bulb, (6) metasense, (7) airbeam, (8) kitchen angular motion, (9) kitchen locator beacon 1, (10) kitchen cabinet contact 1, (11) kitchen cabinet contact 2, (12) kitchen locator beacon 2, (13) kitchen pantry contact, (14) dining room multi-sensor, (15) dining room locator beacon, (16) living room locator beacon 1, (17) living room motion 1, (18) TV smart plug, (19) living room angular motion, (20) living room motion 2, (21) living room locator beacon 2. There are also two data aggregators (smart hubs), one covering the kitchen and one covering living room & dining room, which aggregate data from the different sensors and send it to the cloud. Since this deployment has the main goal of studying edge processing, all the sensors have a Raspberry Pi Zero associated with them which helps in local processing and data routing. In such a heterogeneous deployment, different sensors send different types of data at various sampling frequencies. Sensors such as door contacts, motion sensors do event based sampling, on the other hand, smart plugs, smart bulbs, angular motion sensors, and air quality sensors sample at constant intervals, ranging from 1/10 sec to 5 sec.

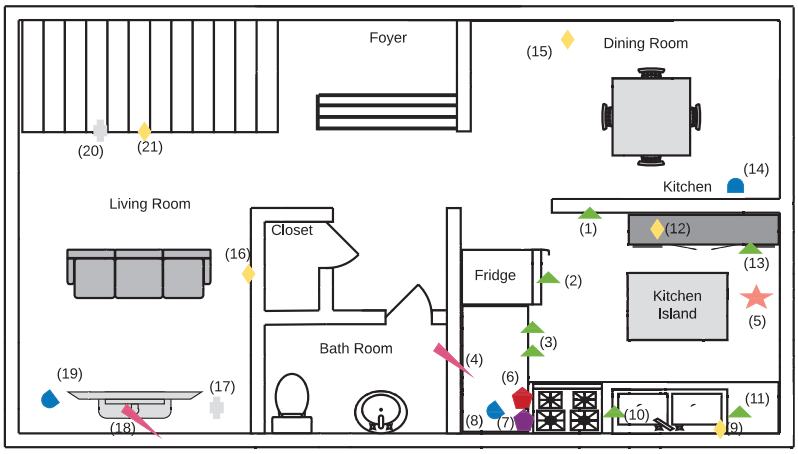


Figure 5.4. Sensor deployment in smart home.

For both experimental scenarios, the coefficients for energy consumption in equation (4) are tuned to fit real sensor hardware according to specifications of the devices. For the battery dynamics model, CF and k parameters are calculated using real ambient temperature data from the temperature sensors. We took $c = 0.5641$ based on the analysis in [108] and set different battery capacities C_R for different devices. The parameters for SoH degradation model is fitted and verified against NREL Li-Ion battery aging dataset [182], and given in Section III.C.

5.4.2 Experimental Results

In this section, we analyze the amount of SoH degradation in the batteries for our proposed method. For comparison, we have selected: i) an "oracle" optimal solution with knowledge of complete horizon, and ii) an optimization method which involves no degradation model and adopts a "linear" energy depletion assumption as presented in [62]. In contrast to our solution, "oracle" is not applicable in practice, but we use it as a benchmark since it gives the optimal solution over the finite horizon. The "linear" solution aims to optimize the network lifetime by minimizing the total energy consumption of the node with maximum energy consumption. This strategy does not consider the dynamics of the battery or the SoH degradation, and essentially tries to optimize the SoC of the batteries. We denote this method as "linear" since it assumes

a linear relation between energy consumption and battery life. We implement all solutions in MATLAB using the YALMIP [183] toolbox.

The "oracle" method requires the knowledge of energy generation, data generation and ambient temperature for all nodes at each time interval in the horizon. The control vector is of size $N \cdot (N-1) \cdot n_T$, so the solution becomes computationally very expensive for large number of nodes N and long time horizons T, where T consists of n_T time steps. Therefore, we did our simulations over a horizon of 1 month, and time intervals of 1 hour. The SoH value of the node with maximum degradation at the end of 1 month horizon for HPWREN and Smart Home scenarios are given in Table 5.3. Smart Home is divided into kitchen & living room because each room has their own smart hub, so the nodes only send data to their respective hubs creating two clusters in the network. Prediction horizon of 6 hours was used for the proposed MPC solution.

Table 5.3. Minimum SoH in the network

	Oracle	Proposed (MPC)
HPWREN	0.9986	0.9983
Smart Home (Kitchen)	0.9987	0.9986
Smart Home (Living Room)	0.9985	0.9985

The overall degradation is very small since the simulation horizon is 1 month (Table 5.3). For the HPWREN experiment, it was observed that the node with the minimum SoH degrades 19.1% more for proposed solution compared to the "oracle". However, the difference is much smaller in Smart Home, where both methods show nearly identical results.

End of battery life simulations

Next, we aim to compare with the "linear" method for the time it takes for the first battery to die. In our solution the prediction horizons of MPC are much smaller than the complete horizon, hence we can simulate through the end of battery life without being restricted by computation resources. Since the "linear" method does not consider energy generation and ambient temperature, it should only know the data generation rates for the whole horizon. In both experiment scenarios we have constant data generation rates which makes the solution

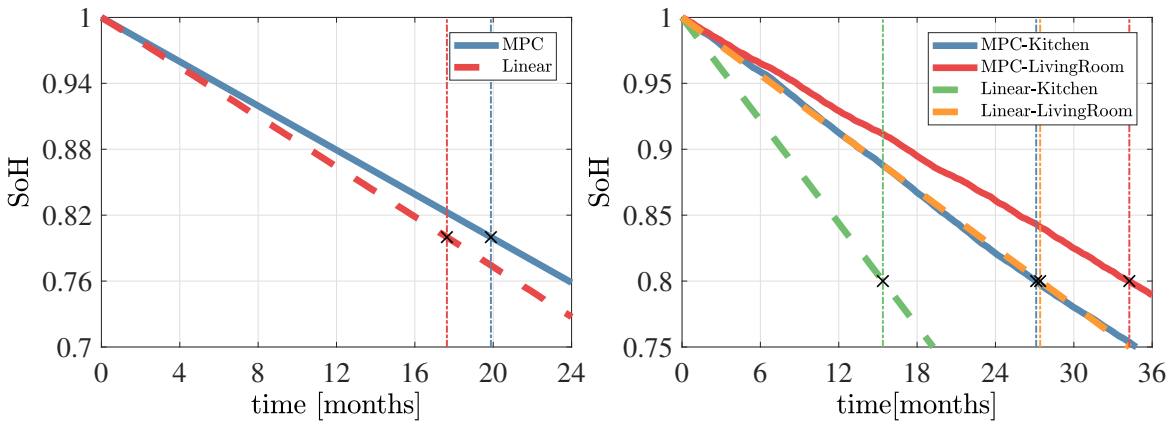


Figure 5.5. Minimum SoH in the network simulated until the end of battery life for HPWREN (left) and Smart Home (right)

time invariant. Therefore, we solve the problem in a short horizon and use the same control vector to simulate until end of battery life. The minimum SoH traces are depicted in Fig. 5.5 for both HPWREN and smart home scenarios. The points where batteries are considered dead ($SoH = 0.8$) shown with vertical lines. By using proposed solution to specifically optimize for SoH of the batteries, we gain 3 months (17.5%) of network lifetime for HPWREN, 11 months (68.7%) for Smart Home (Kitchen), and 7 months (25.0%) for Smart Home (Living Room).

Influence of Prediction Horizon Length

To study the effect of prediction horizon length on the MPC performance, and for the following sections, we simulated a 50-node network distributed randomly in a square region of size 1000m x 1000m, over 1 month horizon. We assume that we have perfect predictions for the given horizon. Fig. 5.6b shows that the MPC solution approaches the "oracle" solution as we increase the prediction horizon. If accurate predictions can be made for the disturbances (e.g. ambient temperature, solar radiation) over a long horizon, this can be leveraged in the MPC to improve SoH by increasing the prediction horizon. However, this heavily depends on the use case. For example, in a smart home the usage patterns of the IoT devices may exhibit high variance because of the human factor. In this case it becomes difficult to predict data generation rates of the nodes, and a short prediction horizon should be used. The computation time per MPC

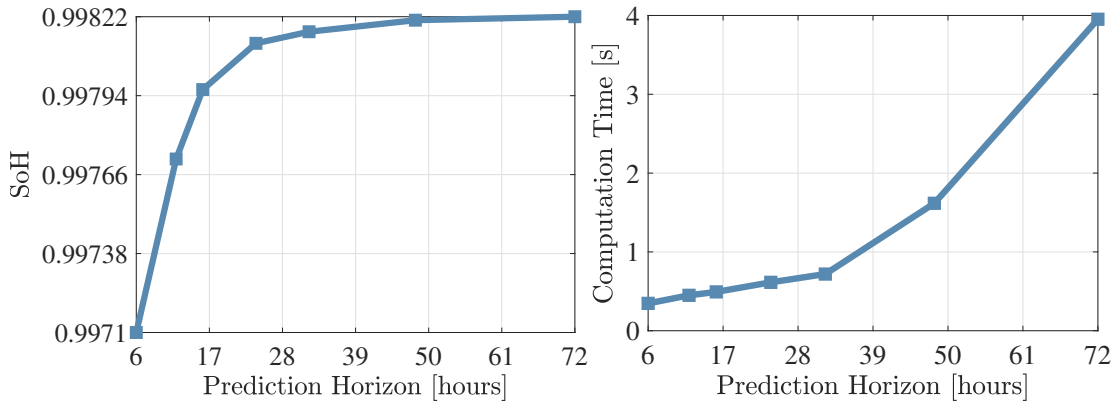


Figure 5.6. Influence of prediction horizon length on SoH degradation (left) and computation time (right)

update step increases as shown in Fig. 5.6b. If an hourly update of the controller is preferred as in our experiments, the increasing computation times does not critically affect the choice of the prediction horizon since they are at least an order of magnitude smaller.

Effect of Ambient Temperature on Battery Health

We compare three cases to analyze the impact of ambient temperature on network life: 1) all nodes are assumed to be under same ambient conditions, 2) nodes have changing ambient temperatures (i.e hourly and daily temperature variations), 3) nodes have constant temperatures, but $\pm 15°C$ temperature difference with respect to each other. The first case is going to be our reference for assessing the impact of ambient temperature. The second case is the closest to a real life scenario, and the third case may also be plausible if there is an altitude difference between the nodes of the network (e.g mountain top), or if there is an obstacle blocking the sun for one node whereas the other node is exposed to direct sunlight.

The first case with the same constant ambient temperatures for all nodes results in 0.104% SoH degradation for the most degraded node. For varying ambient temperature in case 2, the SoH degradation is 0.118% and very close to the constant temperature scenario. Compared to these two cases, having big temperature differences between nodes generates a much faster degrading network with a SoH degradation of 0.143%.

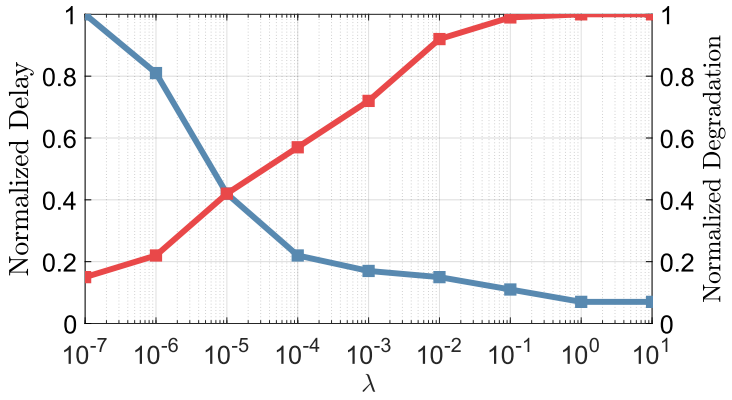


Figure 5.7. Normalized delay and normalized degradation by using end to end delay regularization

End-to-end delay

The normalized delay and degradation results with the additional delay regularization term in the cost function is given in Fig. 5.7. Results show that there is a trade-off between degradation and delay for different values of λ, which controls the level of regularization. Depending on the application needs, this trade-off can be exploited to design IoT network routing schemes with desired performance & lifetime.

5.5 Conclusion

In this chapter, we formulated the problem of minimizing battery degradation to improve the lifetime of IoT networks. We proposed a solution with Model Predictive Control (MPC), leveraging models for battery dynamics and State of Health (SoH). Our work includes the effect of ambient temperature on degradation, and the models we use can accurately capture the nonlinear behavior of actual batteries.

Chapter 5 contains material from "Dynamic Optimization of Battery Heath in IoT Networks", by Kazim Ergun, Raid Ayoub, Pietro Mercati, Tajana Rosing, which appears in International Conference on Computer Design (ICCD), 2019 [4]. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Summary and Future Work

The Internet of Things (IoT) continues to rapidly develop, combining commercial, industrial, residential, and cloud-fog computing domains. It integrates heterogeneous devices, ranging from sensors, to smartphones, tablets and edge servers, and can provide a variety of services, beyond traditional Internet. Unfortunately, due to this unprecedented scale and ubiquity, IoT faces a maintainability challenge and a set of interrelated problems. It has become extremely important for IoT systems to operate reliably for long periods while delivering quality service to users. In this thesis, we focus on novel solutions for energy-efficient and reliability-driven management of IoT systems.

6.1 Thesis Summary

With the emergence of edge computing, IoT devices consume a significant amount of power to deliver high quality of service, which can drain their battery in short time. High peak power increases the device temperature stress, which worsens the impact of transistors and interconnects reliability degradation mechanisms, such as Time Dependent Dielectric Breakdown (TDDB), Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI) and Electromigration (EM). Such mechanisms lead to hard failures and are costly to fix as well as damaging to the manufacturer reputation. These problems worsen with the reduction of transistor and interconnect dimensions due to continuous CMOS technology scaling. Existing management

solutions for IoT systems often neglects these problems.

This thesis proposes the design of an exploration, simulation, and optimization framework for IoT systems and develop end-to-end management solutions to address energy-efficiency and reliability problems. We first introduced a simulation framework called *RelIoT* to enable reliability evaluation and analysis for IoT networks in Chapter 2. The proposed framework is based on the ns-3 simulator and can be used to explore trade-offs between power, performance, and reliability of devices in a network. *RelIoT* is the first framework where these metrics were evaluated simultaneously. We validated our simulations in two experimental network setups. Additionally, we motivated the need for reliability-aware management through example simulation results of energy-optimized and reliability-optimized management strategies. Using *RelIoT*, we can simulate, explore, and check the feasibility of different network configurations in terms of different objectives such as energy efficiency, reliability, and performance. This is of crucial importance as it paves the way for the development of new network management solutions, as shown in Chapter 3 and Chapter 4.

Beyond simulation and exploration, there is a need for proper system design and reliability management strategies at the level of networks of devices. In Chapter 3, 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 is handled in a distributed fashion, resulting in a light-weight and scalable solution. The results of edge computing system simulations indicate that our approach improves the lifetime of a network with 36 edge devices by 5.8 months compared to the closest approach for two gateways to as much as 7.4 months for four 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 and fewer missed task deadlines.

Both computation and communication components play a role in the performance of an IoT service. It is of great importance to include the contributions of both components in reliability analysis and then perform optimization accordingly. In Chapter 4, we explored the problem of maintaining IoT device reliability from the perspective of network routing. The literature on network routing up to date did not study hardware related reliability issues. We proposed a distributed reinforcement learning based routing approach to improve network MTTF, which learns to make its decisions based on the current reliability of the nodes, the amount of degradation they will experience, and networking performance. We extended the ns-3 AODV protocol with a Q-learning algorithm and demonstrated improved network MTTF compared to state-of-the-art methods. Our results show up to a 73.2% improvement in reliability for various communication data rates and the number of nodes in the network while delivering comparable performance.

Finally, in Chapter 5, we formulated the problem of minimizing battery degradation to improve the lifetime of IoT networks. We proposed a solution with Model Predictive Control (MPC), leveraging models for battery dynamics and the State of Health (SoH). Our work includes the effect of ambient temperature on degradation, and the models we use can accurately capture the nonlinear behavior of actual batteries. We demonstrate that our solution can improve network lifetime up to 68.5% compared to conventional energy consumption focused algorithms, which use simple linear battery models. The proposed approach achieves near-optimal performance in terms of preserving battery health, staying within 8.7% SoH with respect to an ideal oracle solution on average.

6.2 Future Directions

Recent years have witnessed machine learning (ML) diffusing into many different domains, including IoT. It has been shown that data-driven, learning based approaches can out-

perform model-based approaches in many applications. Therefore, we are witnessing a lot of opportunities and new business ideas being driven by ML. There has been an increasing effort on both i) developing data-driven control strategies for IoT and ii) using IoT networks for machine learning applications. In this section, we provide future directions for the use of machine learning for IoT and the use of IoT networks for machine learning. We also discuss how to further improve IoT network reliability management solutions. We address the shortcomings, then propose extensions for our dynamic reliability management and routing approaches.

6.2.1 IoT for Machine Learning

Edge devices are equipped with increasingly advanced sensing and computing capabilities. Coupled with advancements in Deep Learning (DL), this opens up countless possibilities for useful applications, e.g., computer vision, robotics, and natural language processing, in domains such as healthcare and vehicular networks. In the traditional cloud-centric DL approaches, data collected by remote clients, e.g., smartphones, is gathered centrally at a computationally powerful server or data center, where the learning model is trained. This requires large volumes of data communicated to the cloud. Communicating massive datasets can result in a substantial burden on the limited network resources. Instead, this process can be split and shared between cloud and edge by leveraging the increased computational capabilities of modern edge devices.

Our computation offloading framework, presented in Chapter 3, 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 internet, which significantly reduces communication costs. Similar to what we proposed in our work, we can optimize computation offloading and find optimal machine learning workload distribution between edge and cloud. The goal is to keep model accuracy high while minimizing training

cost.

Privacy concerns motivate the development of distributed algorithms to allow machine learning at edge networks without data sharing. Federated learning (FL), proposed in [184], has recently drawn significant attention as an alternative to centralized learning. FL exploits the increased computational capabilities of modern edge devices to train a model on the clients' side while keeping their collected data local. In FL, each client performs model training based on its local dataset and shares the model with a central server. The models from all participating clients are then aggregated to a global model. Two of the main challenges associated with FL are the computation and communication bottlenecks [185]. With FL, the computation, i.e., the training process, is pushed to edge devices. However, state-of-the-art ML algorithms, including deep neural networks (DNN), require a large amount of computing power and memory resources to provide better service quality. The DNN models have complicated model architectures with millions of parameters resulting in long training times. Besides computation, the communication load of DNN based FL suffers from the need to repeatedly convey massive model parameters between the server and large number of clients over wireless channels [186]. The solutions proposed in this thesis for efficient computation and communication can be tailored for FL applications for future work. We can use the dynamic management algorithm proposed in Chapter 2 to regulate energy and temperature of edge devices during model training, then refine and leverage our routing approach in Chapter 4 to find optimal paths for communication learning models.

6.2.2 Machine Learning for IoT

Machine learning techniques are recently being used within IoT systems to provide efficient solutions such as localization, clustering, routing, data aggregation, and tasks targeting performance-related challenges, such as congestion control, fault detection, resource management, and security. In particular, the complexity, dynamism, and heterogeneity of modern IoT networks have driven a recent development of routing techniques based on reinforcement

learning (RL) in recent years [138, 139].

We addressed the challenges of IoT networks with a Q-learning based routing approach. However, Q-learning can be computationally costly to implement in large-scale networks due to the growing size of Q-tables. In an ad-hoc network, each node may have to store every next-hop and destination pair for all of the nodes in the network. If the nodes have very large number of neighbors, then approximate Q-learning can be implemented to dramatically reduce the size of Q-tables. By using function approximation, Q-learning can scale to handle very large state-spaces [173]. In particular, deep Q-learning is a promising solution that proposes neural network function approximation for Q-tables [174, 175]. In future work, we want to utilize deep reinforcement learning solutions to overcome such scaling issues.

Our general Q-learning methodology can be adopted and implemented into different protocols, though with modifications, but the specific implementation in our work is based on AODV. For example, a more common routing protocol is RPL [153] for many low-power IoT applications. One of the reasons for using AODV is that it is specifically designed for 'dynamic' networks such a MANETs where the topology change over time. There are efforts on modeling low-power and lossy network protocols such as RPL for IoT, but they are not fully developed yet. Hence, we aim at extending our approach to such protocols in future work.

Another use case of machine learning solutions is in system design. As future work, we plan to leverage our framework for design space exploration (DSE) of IoT networks. Specifically, reinforcement learning can be used for node placement and topology design.

6.2.3 Network Reliability Management

The inherent assumption for our solutions in this thesis is that network Mean Time to Failure (MTTF) is the minimum of any node in the network. This definition is one of the most prevelant in literature [52] and has been used in many recent works [117, 118]. 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 and parallel reliability expressions as described in Chapter 3. Commonly used analytical models for networks include Markov Chains, Fault Tree, Binary Decision Diagram (BDD) and Reliability Block Diagram (RBD). The first step in assessing the impact of individual device reliability and failure mechanisms 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 [187]). Using techniques such as in [188, 189], we can evaluate the system reliability, determine the criticality of IoT devices, and construct system-level models that reflect inter-dependencies between devices.

Bibliography

[1] K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, and T. Rosing, "Simulating reliability of iot networks with reliot," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pp. 25–28, IEEE, 2020.

[2] K. Ergun, R. Ayoub, P. Mercati, and T. Rosing, "Dynamic reliability management of multi-gateway iot edge computing systems," *IEEE Internet of Things Journal*, 2022.

[3] K. Ergun, R. Ayoub, P. Mercati, and T. Rosing, "Reinforcement learning based reliability-aware routing in iot networks," *Ad Hoc Networks*, vol. 132, p. 102869, 2022.

[4] K. Ergun, R. Ayoub, P. Mercati, and T. Rosing, "Dynamic optimization of battery health in iot networks," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, Nov 2019.

[5] S. Li, L. Da Xu, and S. Zhao, "5g internet of things: A survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

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

[7] Cisco Jasper, "The hidden costs of delivering iiot services: Industrial monitoring & heavy equipment," 2016.

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

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

[10] 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.

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

[12] 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.

[13] S. Mittal and J. S. Vetter, "A survey of techniques for modeling and improving reliability of computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 4, pp. 1226–1238, 2015.

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

[15] S. Azzaz and L. Azouz Saidane, "A centralized preventive maintenance strategy for wireless sensor networks," in *Proceedings of the 1st ACM Workshop on High Performance Mobile Opportunistic Systems*, HP-MOSys '12, (New York, NY, USA), p. 61–68, Association for Computing Machinery, 2012.

[16] 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.

[17] "The ns-3 Network Simulator." https://www.nsnam.org/. [Online].

[18] "OMNeT++ Discrete Event Simulator." https://omnetpp.org/. [Online].

[19] X. Chang, "Network Simulations with OPNET," in *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038)*, vol. 1, pp. 307–314, IEEE, 1999.

[20] H. Wu, S. Nabar, and R. Poovendran, "An Energy Framework for the Network Simulator 3 (ns-3)," in *Proceedings of the 4th international ICST conference on simulation tools and techniques*, pp. 222–230, ICST (Institute for Computer Sciences, Social-Informatics and . . . , 2011.

[21] I. Minakov and R. Passerone, "PASES: An Energy-Aware Design Space Exploration Framework for Wireless Sensor Networks," *Journal of Systems Architecture*, vol. 59, no. 8, pp. 626–642, 2013.

[22] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[23] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An Environment for Performance Evaluation of Edge Computing Systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[24] G. Amvrosiadis, "Data storage research vision 2025," *NSF Visioning Workshop*, 2018.

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

[26] "Why edge computing is crucial for the IoT., howpublished=https://www.rtinsights.com/why-edge-computing-and-analytics-is-crucial-for-the-iot/, note = "[online]"."

[27] Y. Xiang, T. Chantem, R. P. Dick, X. S. Hu, and L. Shang, "System-level reliability modeling for mpsocs," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 297–306, 2010.

[28] "Report." http://www.itrs2.net/2013-itrs.html, 2013.

[29] P. Gupta, Y. Agarwal, L. Dolecek, N. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava, *et al.*, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Transactions on Computer-Aided Design of integrated circuits and systems*, vol. 32, no. 1, pp. 8–23, 2012.

[30] G. A. Reis, J. Chang, and D. I. August, "Automatic instruction-level software-only recovery," *IEEE micro*, vol. 27, no. 1, pp. 36–47, 2007.

[31] J. George, B. Marr, B. E. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pp. 158–168, 2006.

[32] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–10, IEEE, 2013.

[33] 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, 2004.

[34] R. Baranowski, A. Cook, M. E. Imhof, C. Liu, and H.-J. Wunderlich, "Synthesis of workload monitors for on-line stress prediction," in *International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, IEEE, 2013.

[35] K. Takeuchi, M. Shimada, S. Konishi, D. Oshida, N. Ota, T. Yasumasu, K. Shibutani, T. Iwashita, T. Kokubun, and F. Tsuchiya, "Experimental implementation of 8.9 kgate stress monitor in 28nm mcu along with safety software library for iot device maintenance," in *2019 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2019.

[36] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Multi-mechanism reliability modeling and management in dynamic systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2008.

[37] X. Xu, K. Teramoto, A. Morales, and H. H. Huang, "Dual: Reliability-aware power management in data centers," in *IEEE/ACM international symposium on cluster, cloud, and grid computing*, 2013.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

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

[54] 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.

[55] 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.

[56] A. Sarkar and T. S. Murugan, "Routing protocols for wireless sensor networks: What the literature says?," *Alexandria Engineering Journal*, vol. 55, no. 4, pp. 3173–3183, 2016.

[57] J. V. Sobral, J. J. Rodrigues, R. A. Rabêlo, J. Al-Muhtadi, and V. Korotaev, "Routing protocols for low power and lossy networks in internet of things applications," *Sensors*, 2019.

[58] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, "A survey on routing metrics," *TIK report*, vol. 262, pp. 1–53, 2007.

[59] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. Costa, O. C. M. Duarte, D. G. Passos, C. V. N. De Albuquerque, D. C. M. Saade, and M. G. Rubinstein, "Routing metrics and protocols for wireless mesh networks," *IEEE network*, vol. 22, no. 1, pp. 6–12, 2008.

[60] "ns3.(2015) Discrete-event network simulator." https://www.nsnam.org/.

[61] L. Canals Casals, M. Rodríguez, C. Corchero, and R. E. Carrillo, "Evaluation of the end-of-life of electric vehicle batteries according to the state-of-health," *World Electric Vehicle Journal*, vol. 10, no. 4, p. 63, 2019.

[62] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 609–619, Aug 2004.

[63] R. Madan and S. Lall, "Distributed algorithms for maximum lifetime routing in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, pp. 2185–2193, Aug 2006.

[64] S. Pourazarm and C. G. Cassandras, "Energy-based lifetime maximization and security of wireless-sensor networks with general nonideal battery models," *IEEE Transactions on Control of Network Systems*, vol. 4, pp. 323–335, June 2017.

[65] 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, pp. 86–98, March 2014.

[66] M. Gatzianas, L. Georgiadis, and L. Tassiulas, "Control of wireless networks with rechargeable batteries [transactions papers]," *IEEE Transactions on Wireless Communications*, vol. 9, pp. 581–593, February 2010.

[67] L. Lin, N. B. Shroff, and R. Srikant, "Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 1021–1034, Oct 2007.

[68] N. Michelusi, L. Badia, R. Carli, L. Corradini, and M. Zorzi, "Energy management policies for harvesting-based wireless sensor devices with battery degradation," *IEEE Transactions on Communications*, vol. 61, pp. 4934–4947, December 2013.

[69] R. Valentini, M. Levorato, and F. Santucci, "Aging aware random channel access for battery-powered wireless networks," *IEEE Wireless Communications Letters*, vol. 5, no. 2, pp. 176–179, 2016.

[70] R. Lajara, J. J. Pérez-Solano, and J. Pelegrí-Sebastiá, "Predicting the batteries' state of health in wireless sensor networks applications," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8936–8945, 2018.

[71] M. Koller, T. Borsche, A. Ulbig, and G. Andersson, "Defining a degradation cost function for optimal control of a battery energy storage system," in *2013 IEEE Grenoble Conference*, pp. 1–6, June 2013.

[72] C. Bordin, H. Anuta, A. Crossland, I. L. Gutierrez, C. Dent, and D. Vigo, "A linear programming approach for battery degradation analysis and optimization in offgrid power systems with solar energy integration.," *Renewable energy.*, vol. 101, pp. 417–430, November 2016.

[73] A. S. Akyurek and T. S. Rosing, "Optimal distributed nonlinear battery control," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 5, pp. 1045–1054, Sep. 2017.

[74] B. Aksanli, T. Rosing, and E. Pettis, "Distributed battery control for peak power shaving in datacenters," in *2013 International Green Computing Conference Proceedings*, pp. 1–8, June 2013.

[75] A. Mamun, I. Narayanan, D. Wang, A. Sivasubramaniam, and H. K. Fathy, "Multi-objective optimization to minimize battery degradation and electricity cost for demand response in datacenters," p. V002T26A004, 10 2015.

[76] "High Performance Wireless Research and Education Network (HPWREN)." http://hpwren.ucsd.edu/.

[77] D. Zhai, R. Zhang, L. Cai, B. Li, and Y. Jiang, "Energy-Efficient User Scheduling and Power Allocation for NOMA-Based Wireless Networks with Massive IoT Devices," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1857–1868, 2018.

[78] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing Resource Allocation in Three-Tier IoT Fog Networks: A Joint Optimization Approach Combining Stackelberg Game and Matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[79] C. Tapparello, H. Ayatollahi, and W. Heinzelman, "Energy Harvesting Framework for Network Simulator 3 (ns-3)," in *Proceedings of the 2nd International Workshop on Energy Neutral Sensing Systems*, pp. 37–42, ACM, 2014.

[80] S. Nikolaev, E. Banks, P. D. Barnes, D. R. Jefferson, and S. Smith, "Pushing the envelope in distributed ns-3 simulations: One billion nodes," in *Proceedings of the 2015 Workshop on Ns-3*, WNS3 '15, (New York, NY, USA), p. 67–74, Association for Computing Machinery, 2015.

[81] S. Nikolaev, P. D. Barnes, J. M. Brase, T. W. Canales, D. R. Jefferson, S. Smith, R. A. Soltz, and P. J. Scheibel, "Performance of distributed ns-3 network simulator," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, (Brussels, BEL), p. 17–23, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.

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

[83] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 105–114, ACM, 2010.

[84] 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.

[85] J. H. Stathis, "Physical and Predictive Models of Ultrathin Oxide Reliability in CMOS Devices and Circuits," *IEEE Transactions on Device and Materials Reliability*, vol. 1, no. 1, pp. 43–59, 2001.

[86] "MQTT MQ Telemetry Transport." https://mqtt.org/. [Online].

[87] 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*, pp. 1–1, 2019.

[88] "Hioki3334 Powermeter." https://www.hioki.com/en/products/detail/?product_key=5812. [Online].

[89] "INA219 Datasheet." http://www.ti.com/lit/ds/symlink/ina219.pdf/. [Online].

[90] "DHT22 Datasheet." https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf/. [Online].

[91] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pp. 1–14, 2017.

[92] 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.

[93] Jae-Hwan Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, pp. 609–619, Aug 2004.

[94] 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.

[95] 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.

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

[97] 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.

[98] 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.

[99] 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.

[100] 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.

[101] 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.

[102] 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.

[103] 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.

[104] 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.

[105] 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)*, pp. 960–965, IEEE, 2015.

[106] 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.

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

[108] 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.

[109] 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.

[110] 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.

[111] 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.

[112] 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.

[113] 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.

[114] 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.

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

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

[117] 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.

[118] 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.

[119] G. Torrisi, *Low-complexity numerical methods for nonlinear model predictive control*. PhD thesis, ETH Zurich, 2017.

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

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

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

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

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

[125] 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.

[126] 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.

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

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

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

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

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

[132] 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.

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

[134] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, 2007.

[135] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "Residual energy-based cluster-head selection in wsns for iot application," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5132–5139, 2019.

[136] A. Dhumane, R. Prasad, and J. Prasad, "Routing issues in internet of things: a survey," in *Proceedings of the international multiconference of engineers and computer scientists*, 2016.

[137] M. Zhao, I. W.-H. Ho, and P. H. J. Chong, "An energy-efficient region-based rpl routing protocol for low-power and lossy networks," *IEEE Internet of Things Journal*, vol. 3, no. 6, 2016.

[138] Z. Mammeri, "Reinforcement learning based routing in networks: Review and classification of approaches," *IEEE Access*, vol. 7, pp. 55916–55950, 2019.

[139] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM workshop on hot topics in networks*, 2017.

[140] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp. 134–146, 2003.

[141] NREL, "National Solar Radiation Database (NSRDB)." https://maps.nrel.gov/nsrdb-viewer. [Online].

[142] K. Ergun, R. Ayoub, P. Mercati, and T. Rosing, "Improving mean time to failure of iot networks with reliability-aware routing," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4, IEEE, 2021.

[143] M. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *4th international workshop on mobile and wireless communications network*, pp. 368–372, IEEE, 2002.

[144] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," 2001.

[145] D. Zhang, G. Li, K. Zheng, X. Ming, and Z.-H. Pan, "An energy-balanced routing method based on forward-aware factor for wireless sensor networks," *IEEE transactions on industrial informatics*, vol. 10, no. 1, pp. 766–773, 2013.

[146] S. B. Shah, Z. Chen, F. Yin, I. U. Khan, and N. Ahmad, "Energy and interoperable aware routing for throughput optimization in clustered iot-wireless sensor networks," *Future Generation Computer Systems*, vol. 81, pp. 372–381, 2018.

[147] R. Ullah, Y. Faheem, and B.-S. Kim, "Energy and congestion-aware routing metric for smart grid ami networks in smart city," *IEEE access*, vol. 5, pp. 13799–13810, 2017.

[148] A. A. Rahat, R. M. Everson, and J. E. Fieldsend, "Hybrid evolutionary approaches to maximum lifetime routing and energy efficiency in sensor mesh networks," *Evolutionary computation*, vol. 23, no. 3, pp. 481–507, 2015.

[149] X. Li, B. Keegan, F. Mtenzi, T. Weise, and M. Tan, "Energy-efficient load balancing ant based routing algorithm for wireless sensor networks," *IEEE Access*, 2019.

[150] Z. Wadud, N. Javaid, M. A. Khan, N. Alrajeh, M. S. Alabed, and N. Guizani, "Lifetime maximization via hole alleviation in iot enabling heterogeneous wireless sensor networks," *Sensors*, vol. 17, no. 7, p. 1677, 2017.

[151] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, IEEE, 1999.

[152] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol (olsr)," 2003.

[153] T. Winter, P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J.-P. Vasseur, R. K. Alexander, *et al.*, "Rpl: Ipv6 routing protocol for low-power and lossy networks.," *rfc*, vol. 6550, pp. 1–157, 2012.

[154] E. Yitayal, J.-M. Pierson, and D. Ejigu, "A balanced battery usage routing protocol to maximize network lifetime of manet based on aodv," in *International Conference on Next Generation Wired/Wireless Networking*, pp. 266–279, Springer, 2014.

[155] P. O. Kamgueu, E. Nataf, T. D. Ndié, and O. Festor, "Energy-based routing metric for rpl," 2013.

[156] J.-M. Kim and J.-W. Jang, "Aodv based energy efficient routing protocol for maximum lifetime in manet," in *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pp. 77–77, IEEE, 2006.

[157] O. Iova, F. Theoleyre, and T. Noel, "Improving the network lifetime with energy-balancing routing: Application to rpl," in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–8, IEEE, 2014.

[158] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, pp. 671–678, 1994.

[159] W. Naruephiphat and W. Usaha, "Balanced energy-efficient routing in manets using reinforcement learning," in *2008 International Conference on Information Networking*, pp. 1–5, IEEE, 2008.

[160] D. Macone, G. Oddi, and A. Pietrabissa, "Mq-routing: Mobility-, gps-and energy-aware routing protocol in manets for disaster relief scenarios," *Ad Hoc Networks*, vol. 11, no. 3, pp. 861–878, 2013.

[161] Y.-H. Chang, T. Ho, and L. P. Kaelbling, "Mobilized ad-hoc networks: A reinforcement learning approach," in *International Conference on Autonomic Computing, 2004. Proceedings.*, pp. 240–247, IEEE, 2004.

[162] C. Wu, K. Kumekawa, and T. Kato, "A manet protocol considering link stability and bandwidth efficiency," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pp. 1–8, IEEE, 2009.

[163] K. Wang, W.-C. Wong, and T. Y. Chai, "A manet routing protocol using q-learning method integrated with bayesian network," in *2012 IEEE International Conference on Communication Systems (ICCS)*, pp. 270–274, IEEE, 2012.

[164] S. Chettibi and S. Chikhi, "Dynamic fuzzy logic and reinforcement learning for adaptive energy efficient routing in mobile ad-hoc networks," *Applied Soft Computing*, vol. 38, pp. 321–328, 2016.

[165] A. Kozłowski and J. Sosnowski, "Energy efficiency trade-off between duty-cycling and wake-up radio techniques in iot networks," *Wireless Personal Communications*, 2019.

[166] K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, and T. Rosing, "Reliot: Reliability simulator for iot networks," in *Internet of Things - ICIOT*, 2020.

[167] A. Grenat, S. Sundaram, S. Kosonocky, R. Rachala, S. Sambamurthy, S. Liepe, M. Rodriguez, T. Burd, A. Clark, M. Austin, *et al.*, "4.2 increasing the performance of a 28nm x86-64 microprocessor through system power management," in *ISSCC*, IEEE, 2016.

[168] P. Mercati, A. Bartolini, F. Paterna, T. S. Rosing, and L. Benini, "Workload and user experience-aware dynamic reliability management in multicore processors," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, IEEE, 2013.

[169] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[170] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[171] N. J. Jevtic and M. Z. Malnar, "Novel etx-based metrics for overhead reduction in dynamic ad hoc networks," *IEEE Access*, 2019.

[172] H. Li and D. Yu, "A statistical study of neighbor node properties in ad hoc network," in *Proceedings. International Conference on Parallel Processing Workshop*, pp. 103–108, IEEE, 2002.

[173] D. Pandey and P. Pandey, "Approximate q-learning: An introduction," in *2010 second international conference on machine learning and computing*, pp. 317–320, IEEE, 2010.

[174] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for Dynamics and Control*, pp. 486–489, PMLR, 2020.

[175] P. Xu and Q. Gu, "A finite-time analysis of q-learning with neural network function approximation," in *International Conference on Machine Learning*, pp. 10555–10565, PMLR, 2020.

[176] "Santa Margarita Ecological Reserve (SMER)." https://fsp.sdsu.edu//.

[177] S. Chettibi and S. Chikhi, "Adaptive maximum-lifetime routing in mobile ad-hoc networks using temporal difference reinforcement learning," *Evolving Systems*, vol. 5, no. 2, pp. 89–108, 2014.

[178] Mark Hung, Gartner Research, "Leading the IoT." https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf, 2017. [Online].

[179] "Jasper.(2016)The Hidden Costs of Delivering IIoT Services." https://www.cisco.com/c/dam/m/en_ca/never-better/manufacture/pdfs/hidden-costs-of-delivering-iiot-services-white-paper.pdf.

[180] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*, vol. 2. prentice hall PTR New Jersey, 1996.

[181] A. Hoke, A. Brissette, D. Maksimovic, A. Pratt, and K. Smith, "Electric vehicle charge optimization including effects of lithium-ion battery degradation," in *2011 IEEE Vehicle Power and Propulsion Conference*, pp. 1–8, Sept 2011.

[182] K. Goebel, B. Saha, A. Saxena, J. R. Celaya, and J. P. Christophersen, "Prognostics in battery health management," *IEEE Instrumentation Measurement Magazine*, vol. 11, pp. 33–40, August 2008.

[183] J. Lofberg, "Yalmip : a toolbox for modeling and optimization in matlab," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, pp. 284–289, Sep. 2004.

[184] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017.

[185] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, 2021.

[186] T. Li *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020.

[187] 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.

[188] 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.

[189] 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*, pp. 274–282, IEEE, 2013.