ENABLING RELIABLE AND EFFICIENT DATA TRANSFER FOR INTERNET OF THINGS APPLICATIONS

BY

DI MU

M.S., New Jersey Institute of Technology, 2012
B.S., Hangzhou Dianzi University, 2010

DISSERTATION

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2021

Accepted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Computer Science
in the Graduate School of
Binghamton University
State University of New York
2021

July 7, 2021

Abstract

We are now in the golden age of the Internet of Things (IoT) and various IoT applications are being developed every day. Research efforts over the last few decades produced multiple wireless technologies, which are readily available to support communication between devices in various IoT applications. However, it is very challenging to achieve reliable and energy-efficient data transfer through wireless medium due to its inherent unreliability. The data transfer challenge is significantly amplified by the uncertainties of network traffic and wireless environments, the real-time requirement of data deliveries, the high mobility of IoT platforms, and the high network management complexity. Fortunately, heterogeneous radios and new wireless technologies are becoming increasingly available on modern IoT devices, offering new opportunities to address the data transfer challenge. This dissertation presents my research that leverages those new opportunities to enable reliable and energy-efficient data transfer for IoT applications. To address the uncertainties of network traffic and wireless environments, we develop the Adaptive Radio and Transmission Power Selection (ARTPoS) system that leverages multiple heterogeneous radios and selects the most suitable radio(s) and transmission power(s) for the current conditions and requirements. To support real-time data deliveries, we develop the Real-time Radio Switching and Bundling (RRaSB) system that runs on our embedded platform equipped with multiple heterogeneous radios and allows dynamic radio switching and bundling among them. To accommodate the needs of mobile IoT platforms, we develop the ShuttleNet system that collects real-time data from running vehicles based on the Low-Power Wide-Area Network (LPWAN) technology and selects the network's physical-layer parameters at runtime. To reduce network management complexity, we develop the Direct Message Dissemination (DIME) system that leverages the Cross-Technology Communication (CTC) technique to efficiently time synchronize wireless devices in the network and enable direct message deliveries through long distance links instead of hop-by-hop transports. Real-world and testbed experiments show that our designs provide significant improvements over the state of the art.

Dedicated to my parents.

Acknowledgements

The dissertation would not have been possible without the support, guidance, and advice of my advisor and colleagues. I would like to first thank my advisor, Dr. Mo Sha, for his patient guidance and unwavering support throughout my doctoral research. Dr. Mo Sha has given me great support and valuable advice in all aspects of my research. I also appreciate my dissertation committee members Dr. Kyoung-Don Kang, Dr. Yifan Zhang, and Dr. Yu Chen for their important feedback and guidance on my dissertation. I would also like to thank my colleagues Junyang Shi, Xia Cheng, Yitian Chen, Xingjian Chen, Hyungdae Yi, Yunpeng Ge, and Bin Wang for their dedicated work and assistance.

Important acknowledgements should be given to the organizations and people that supported the research projects. I thank the faculty and staff in the Computer Science Department, especially our IT specialists Mr. Robert L Mess and Mr. Dave Hall who assisted us in deploying and maintaining the LoRa base station and the wireless network testbed. I also thank the Transportation and Parking Services at SUNY-Binghamton for offering the platform for LoRa end devices. My research was partially supported by the NSF through grant CRII-1657275 (NeTS). The Graduate School and the Graduate Student Employees' Union at SUNY-Binghamton provided travel funds for conferences.

Last but not least, I would like to thank my family for their constant encouragement and understanding.

Contents

List of Tables

List of Figures

1 Introduction

We are now in the golden age of the Internet of Things (IoT) and various IoT applications are being developed every day. Research efforts over the last few decades produced multiple wireless technologies, which are readily available to support communication between devices in various IoT applications. However, it is very challenging to achieve reliable and energy-efficient data transfer through wireless medium due to its inherent unreliability. The data transfer challenge is significantly amplified by the uncertainties of network traffic and wireless environments, the real-time requirement of data deliveries, the high mobility of IoT platforms, and the high network management complexity. Failing to achieve reliable and energy-efficient data transfer may result in degraded Quality of Service (QoS), extra maintenance cost, or safety hazard. Fortunately, heterogeneous radios and new wireless technologies are becoming increasingly available on modern IoT devices, offering new opportunities to address the data transfer challenge.

Most current IoT applications use a single wireless technology to support data transfer. However, each wireless technology is originally designed with different goals, such as high throughput, low power consumption, and robustness to interference. Using a single wireless technology therefore cannot deliver optimal performance under varying network traffic and/or changing wireless environment. To adapt to the uncertainties of network traffic and wireless environment, we develop the Adaptive Radio and Transmission Power Selection (ARTPoS) system that leverages multiple heterogeneous radios and selects the most suitable radio(s) and transmission power(s) for the current conditions and requirements [1, 2].

The importance of real-time wireless data transfer is increasing for IoT applications in recent years, demanding real-time and energy-efficient communication through wireless medium. However, it is challenging to support stringent timing constraints energy-efficiently through wireless medium due to its inherent unreliability and timing-unpredictability. To support real-time data deliveries energy-efficiently, we develop the Real-time Radio Switching and Bundling (RRaSB) system that runs on our embedded platform equipped with multiple heterogeneous radios and allows dynamic radio switching and bundling among them based on our real-time radio selection and data partitioning algorithms [3, 4].

Recently, the Low-Power Wide-Area Network (LPWAN) technologies, such as LoRa, have been used as low-cost alternatives that provide the capability of long-range data collection, in replace of satellite and cellular technologies that have been traditionally used. However, the mobility of IoT platforms (e.g., running vehicles) poses a significant challenge to make good tradeoffs between network reliability and throughput under the fluctuations of link quality resulting from the mobility. To support mobile IoT platforms on LoRa-based wireless networks, we develop a campus shuttle monitoring system, namely ShuttleNet, which collects real-time data from running vehicles through LoRa links and controls the network's physical-layer parameters at runtime based on the current link conditions [5].

Industrial wireless networks, which typically connect sensors, actuators, and controllers in industrial facilities, have critical demands for reliable and real-time communication. However, those networks often suffer high network management complexity due to a large number of network devices and a mesh network topology. To reduce the network management complexity, we develop the Direct Message Dissemination (DIME) system that leverages the Cross-Technology Communication (CTC) technique to efficiently time synchronize wireless devices in the network and enable direct message deliveries through long distance links instead of hop-by-hop transports.

The rest of the dissertation is organized as follows. Chapter 2 reviews the related work. Chapter 3 introduces our design of ARTPoS. Chapter 4 presents our RRaSB system. Chapter 5 presents our ShuttleNet system. Chapter 6 discusses our design of DIME. Chapter 7 concludes this dissertation.

2 Related Work

2.1 Runtime Radio Selection and Transmission Power Control

In recent years, heterogeneous radio technologies are advancing fast and many new IoT platforms are equipped with multiple heterogeneous radios. Bandwidth aggregation for a device with multiple network interfaces has been studied extensively in the literature and many techniques are readily available [6]. For instance, multipath TCP (MPTCP) [7] is one of the most widely used techniques and now a new standardized transport protocol that allows a device to take advantage of data transfer through multiple network interfaces simultaneously. Those early efforts are not directly applicable to embedded wireless devices with power constraints, since they were not designed to provide energy-efficient wireless data transfers [8, 9]. There has also been increasing interest in studying the energy-aware bundling or switching between WiFi and 3G/4G radios on smartphones [10, 11]. For instance, Bui et al. used WiFi and/or LTE to minimize playback halts due to the buffer underflow when a video is streamed to a smartphone [11]. There exists software, e.g., VideoBee, Super Download Lite-Booster, MPTCP in iOS, KT's GiGA LTE, that support concurrent use of WiFi and cellular radios. These existing approaches are either unaware of transmission power control or limited to mainly WiFi and 3G/4G radios on smartphone platforms, thus they are not directly applicable to support energy-efficient data transfer using heterogeneous radios in various IoT embedded platforms. Generally speaking, it is largely unknown how to energy-efficiently use radios with very different characteristics through runtime radio and transmission power adaptation. To address this critical gap in the current state of the art, we investigate the joint impact of radio and transmission power selection on energy efficiency and link reliability, and propose a practical approach that intelligently uses a high throughput radio (i.e., WiFi) and an energy-efficient radio (i.e., ZigBee) in Chapter 3. To our knowledge, our ARTPoS system is the first to support not only runtime bundling and switching between WiFi and ZigBee but also adaptive transmission power control, that proactively minimizes power consumption subject to given network traffic and operating conditions.

Transmission power control for a single radio has been extensively investigated in the literature of wireless sensor networks and wireless mesh networks. Indirect link quality metrics such as received signal strength (RSS) and link quality indicator (LQI) [12, 13] or direct link quality metrics such as packet reception ratio (PRR) and packet error rate (PER) [14, 15] have been used to measure the link quality. Heuristics [14, 16, 17] and control-theoretic approaches [13, 12, 15] have been applied to achieve the desirable link quality by controlling the transmission power at runtime. These existing approaches, designed to select the transmission power of a single radio, are not directly applicable for heterogeneous radios, since the power consumption has to be compared between different radios and the link quality and power consumption of multiple radios have to be jointly considered. In contrast, ARTPoS employs a pragmatic integrated systems approach to optimize the transmission power selection together with the radio selection.

2.2 Energy-Efficient Real-Time Data Transfer Using Heterogeneous Radios

More recently, research efforts have begun to pay more attention to the energy efficiency of data transfers using heterogeneous radios in the context of smartphones and IoT applications. For instance, Lim et al. extended MPTCP to support energy-aware data transfers over WiFi and LTE radios [18]. Nikraves et al. conducted a real-world study of multipath for mobile settings and developed a flexible software architecture to enhance the performance of MPTCP on smartphones [8]. Nika et al. developed an energy model for smartphones to support energy-aware WiFi and LTE radio bundling [19]. Wu et al. designed an energy-efficient WiFi and LTE bandwidth aggregation method for video services on mobile devices [20]. Gu et al. developed a low-power out-of-band control plane based on LoRa for multi-hop ZigBee networks [21]. These existing approaches are either unaware of the timing constraints of data deliveries or limited to only WiFi and 3G/4G radios on smartphone platforms, thus they are not directly applicable to support real-time data transfer using heterogeneous radios in various IoT embedded platforms.

For real-time wireless data deliveries, novel methods (e.g., [22, 23, 24]) have recently been explored to meet timing constraints via real-time MAC protocols, packet scheduling, and routing based on the centralized Time Division Multiple Access (TDMA) scheme. However, most of them consider neither energy efficiency nor heterogeneous radios. In contrast to these real-time approaches, we propose our RRaSB system in Chapter 4 that aims to support stringent timing constraints with minimal energy consumption by effectively leveraging heterogeneous radios.

Our work is therefore orthogonal and complementary.

2.3 Long-Range Vehicular Data Collection and Spreading Factor Control

Satellite and cellular technologies are traditionally used to collect real-time data from moving vehicles through long-distance links. For instance, vehicular satellite links have been used to connect emergency vehicles to information headquarters in disaster areas [25], while LTE-based communication systems have been integrated into the urban transit systems [26, 27]. LTE-based vehicle-to-everything (V2X) communication is currently being standardized by 3GPP [28]. Unfortunately, those satellite or cellular based systems are often very costly because they use expensive devices and licensed frequency bands, which limits their applications. In recent years, LoRa, which is an emerging Low-Power Wide-Area Network (LPWAN) technology, has been used as a low-cost alternative that provides the capability for long-range data collection to low data rate applications. For instance, Islam et al. proposed a LoRa link scheduling algorithm and tested it in city environments [29]. Liando et al. conducted large-scale measurements on the performance of a campus-wide LoRa network and studied the impact of LoRa transmission parameters on link performance [30]. More recently, LoRa has been employed to support vehicular communication. For example, Santa et al. developed a LoRa-based vehicular monitoring platform [31]. Salazar-Cabrera et al., Boshita et al., and Guan et al. presented prototypes of public vehicle tracking systems that use LoRa to transfer the real-time locations and operating conditions of vehicles [32, 33, 34]. Bertoldo et al. deployed LoRa end devices on public transportation vehicles to transfer environmental sensor readings [35]. Ouya et al. proposed a LoRa-based communication protocol that allows electric vehicles and charging stations to exchange information on energy demand and availability [36]. The existing studies have demonstrated the feasibility of using LoRa to reliably collect data over long distances. The spreading factor (SF), which is a physical layer parameter of LoRa that determines the transmission data rate, plays an important role in the tradeoff between the reliability and throughput of the data collection from running vehicles. As presented in Chapter 5, our yearlong empirical study provides valuable insights on selecting SF configurations for the LoRa end devices with mobility. To our knowledge, our ShuttleNet system is the first to investigate the SF selection for LoRa end devices installed on running vehicles, distinguished from previous work using static SF configurations.

In the literature, several approaches have been proposed to configure SF for LoRa networks based on link quality. For instance, the ADR algorithm, specified in LoRaWAN, estimates the link quality using the maximum SNR in 20 historical samples and selects the SF configuration based on the required SNR for each SF [37]. ADR+ is an enhanced version of ADR which replaces the maximum SNR with the average SNR to estimate the link quality [38]. The Probing algorithm makes use of the measured PRR to configure SF and gradually approaches the optimal configuration [39]. Unfortunately, those SF control approaches designed for stationary LoRa end devices do not work well for mobile devices. In Chapter 5, our runtime SF control solution significantly outperforms the existing solutions. There also exist some approaches that estimate the link quality and select the SF configuration based on the GPS locations of LoRa end devices [40, 41, 42].

However, those GPS-based approaches significantly increase the system cost and power consumption of LoRa end devices. Our experimental results of ShuttleNet show that the measured link characteristics can be used reliably to select good SF configurations and there is no need to install those GPS devices.

KNN is a classical machine learning technique that uses the nearest neighbors in the collected data set to determine the class or value of a query example [43]. KNN has been demonstrated as an efficient and effective algorithm when applied to solve many wireless communication problems. For instance, Yu et al. and Arya et al. used KNN for indoor and outdoor localization based on RSS measurements, respectively [44, 45]. Li et al. employed KNN to detect network intrusions in Wireless Sensor Networks (WSNs) [46] while Pan et al. used KNN to estimate the missing data during data transfers in WSNs [47]. Donohoo et al. used KNN to predict the energy demand of mobile devices [48]. Ma et al. employed KNN to predict the PRR of 802.11 links based on SNR measurements [49]. To our knowledge, ShuttleNet is the first to use KNN to select SF configuration for mobile LoRa end devices. Our experimental results demonstrate the effectiveness and efficiency of our proposed solution.

2.4 Industrial Wireless Network Management

In 2012, IEEE 802.15.4e introduced Time-Slotted Channel Hopping (TSCH) [50], which is a Medium Access Control (MAC) protocol in the context of Low-Power and Lossy Networks (LLNs). Industrial wireless networks employ TSCH to provide time-deterministic packet deliveries for industrial process control and automation. TSCH networks are globally time synchronized to support time-slotted access. Periodic synchronization beacons are propagated from the coordinator node to every node in the network through hop-by-hop transports. However, TSCH networks often suffer high network management complexity due to a large number of network nodes and a mesh network topology. For example, every TSCH node needs to transmit a synchronization beacon periodically, which is a significant network management overhead. Also, TSCH networks often take a long time at startup to synchronize the entire network and stabilize the network performance.

Research efforts in recent years have proposed autonomous scheduling solutions for TSCH-based mesh networks, which aim to enhance the scalability of those networks by removing the centralized scheduler. For instance, Orchestra [51] allows TSCH nodes to maintain their schedules autonomously based on the state of the routing protocol. ALICE [52] is an autonomous link-based cell scheduling scheme which allocates a unique cell for each directional link by using the local information in the routing layer. DiGS [53] is a distributed graph routing and autonomous scheduling solution that allows the field devices to compute their own transmission schedules based on the graph routes. However, these efforts cannot completely resolve the issue of high network management complexity, since the deliveries of critical network management data, such as time information, control commands, and urgent alerts, still rely on hop-by-hop transports and thus may suffer insufficient reliability and undesirable latency.

Emerging LPWAN technologies, such as LoRa, offer new opportunities to address the high network management complexity by using one-hop long-distance links. The LoRaCP system [21] features one-hop out-of-band control planes to deliver control messages in ZigBee mesh networks based on additional LoRa radios. However, adding new radio modules to existing hardware platforms may increase the cost and complexity of deployment and maintenance. Fortunately, new Cross-Technology Communication (CTC) techniques have enabled direct messaging from an LPWAN radio to 802.15.4-based field devices without any hardware change on the field devices. For instance, Shi et al. proposed two physical-layer CTC approaches to enable ZigBee devices to decode the information carried by LoRa transmissions in the 2.4 GHz and Sub-GHz bands [54, 55]. In Chapter 6, we propose the DIME system that leverages the long-distance links provided by the CTC technique to overcome the high management complexity in TSCH-based autonomously scheduled networks. Our work is therefore orthogonal and complementary.

3 Robust Optimal Selection of Radio Type and Transmission Power for Internet of Things

3.1 Introduction

Diverse wireless technologies, produced by research over the years, are available to support communication between devices in various Internet of Things (IoT) applications. However, each of these technologies were originally designed with different goals, such as high throughput, low power consumption, low latency, and robustness to interference, and thus offer very different characteristics. Single radio technology can hardly deliver optimal performance in all desirable quality of service (QoS) dimensions, especially under varying environmental conditions. For instance, WiFi can provide high throughput, but suffers from high power consumption. A considerable amount of energy can be wasted if a WiFi radio experiences irregular data transmission at low data rate such that it stays longer in a power-hungry active mode, rather than in the power save mode. On the other hand, ZigBee is power-efficient, but cannot support high data rate applications.

Using a single wireless technology therefore cannot meet the demands of varying workloads or changing environmental conditions. This issue becomes further pronounced with emerging mobile IoT applications that involve placing embedded devices on the user's body or other mobile objects. Monitoring and controlling mobile objects open up opportunities for novel and exciting IoT applications (e.g., assisted living, health monitoring, and multi-agent autonomous vehicular and robotic systems), while also introducing the fundamental challenge of maintaining optimal wireless communication between devices under the following uncertainties: **Network Traffic Uncertainties**: The network traffic is subject to spontaneous changes. For instance, in a health monitoring application, a wearable device may produce low amount of data during some hours of the day, but sporadically require rapid transmission of large volume of data in response to a critical medical condition. Moreover, devices may have multiple sensors, with diverse traffic patterns, and the system may turn ON or OFF any of the sensors at any given time [56]. **Wireless Environment Uncertainties**: The wireless environment changes when the device moves around. At times, a mobile device will need to be able to deal with a highly noisy environment; at other times it may enjoy a clean environment [56]. A stationary device may also experience environment changes due to changing ambient interference. Given the dynamic nature of communication in IoT applications, a traditional one-radio-fits-all approach cannot meet the challenges associated with the dynamics and uncertainties in network traffic and operating conditions.

Fortunately, embedded system hardware and radio technologies have been seeing appreciable advancement. Heterogeneous radios, e.g., WiFi, LTE, Bluetooth, and ZigBee are becoming increasingly available in modern embedded or mobile devices. Most smartphones nowadays support WiFi, LTE, and Bluetooth. A majority of modern devices designed for IoT applications also support heterogeneous radios. For instance, Firestorm platform [57] supports Bluetooth low energy (BLE) and ZigBee and uses a 32 bit low-power microcontroller with the duty cycling capability. TI CC2650 [58] integrates two radios (i.e., ZigBee and BLE) on a single chip. Raspberry Pi 3 model B [59] uses a Broadcom single-chip radio supporting both WiFi and BLE. IOT-Gate-iMX7 [60] is an industrial IoT gateway, which supports 4G/LTE, WiFi, Bluetooth, and Zigbee. The ZiFi device [61] support both WiFi and ZigBee. Recent hardware advancement offers new opportunities to use multiple wireless technologies efficiently.

This chapter aims to address the previously stated networking challenges, while leveraging the above-stated hardware advancements; specifically, it makes the following contributions:

- We design the *Adaptive Radio and Transmission Power Selection (ARTPoS)* system that makes available multiple wireless technologies at runtime and selects the radio(s) and their transmission power(s) that are best suited for the current network traffic and operating conditions.

- We develop new offline modeling approaches that allow the selection system to **adapt** to large variance in power consumption and link reliability measurements.

- We formulate the problem of radio and transmission power selection as an optimization problem[1] and develop two practical (lightweight) online solutions; the latter solution uniquely allows online updating of the link reliability models, to enable adapting to runtime environments that deviate from the offline settings that were used to train the models.

- We implement the ARTPoS in Raspbian Linux and Contiki and evaluate

[1]We focus on minimizing the energy consumption on the link level and the sender side (IoT end devices), since the IoT gateways are usually not or much less energy-constrained.

it on a new embedded platform supporting WiFi, ZigBee, and BLE; these efforts demonstrate the unique benefits of **adaptive** runtime selection of radios and their transmission powers.

We show that our ARTPoS implementations clearly outperform two baselines (Fixed-power and ART-WiFi) in terms of power consumption, while delivering similar link reliability. Expectedly, ART-ZigBee registers the lowest power consumption, but fails to provide any meaningful link reliability for all data rates above 1000 packets/period. Importantly, this advantage of the ARTPoS implementations is shown to hold under various indoor and outdoor settings, and with and without interference. Lastly, we show that the newer ARTPoS-irp version is able to exploit its special (runtime) model adaptation capacity to provide, on average, a 3.7% better packet delivery rate and $13.5mW$ power savings, over the original ARTPoS implementation.

The remainder of the chapter is organized as follows. Section 3.2 introduces our ARTPoS design. Section 3.3 presents the power consumption and link reliability modeling and Section 3.4 introduces our problem formulation and solution strategy. Section 3.5 presents our experimental evaluation. Section 3.6 concludes the chapter.

3.2 ARTPoS System Architecture

This section presents the design of ARTPoS. Fig. 3.1 shows the system architecture. The **Modeling Engine** generates the power consumption and link reliability models needed for the radio and transmission power selection (Section 3.2.1). The **Radio/Transmission Power Selection Engine** selects the

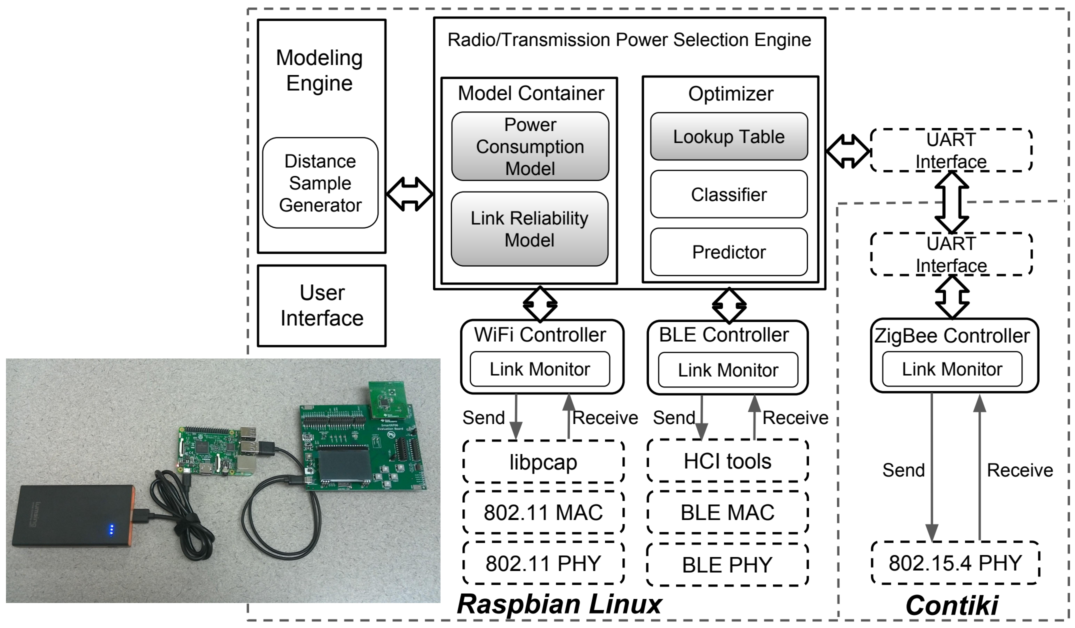


Figure 3.1: System architecture.

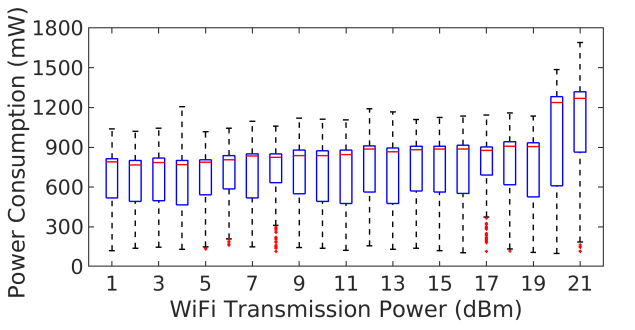
best-suited radio(s) and transmission power(s) based on the application specified data rate and the throughput of each available link measured at runtime (Section 3.2.2). Multiple **Radio Controller** modules (e.g., WiFi, BLE, and ZigBee controllers) exist in ARTPoS. Each radio controller controls the state (i.e., On or Off) of a radio and sets its transmission power based on the decision made by the Radio/Transmission Power Selection Engine, while the **User Interface** supports the interactions with system users (Section 3.2.3).

To support the realization of ARTPoS, we have built a new embedded platform (as shown in Fig. 3.1) with heterogeneous radios consisting of WiFi, ZigBee, and BLE by instrumenting a Raspberry Pi 3 Model B [59] with a TI CC2650 Development Kit [58], which is connected to the Raspberry Pi through a USB port. Raspberry Pi integrates a Broadcom BCM43438 single chip radio processor supporting WiFi and BLE, while CC2650 is the core wireless MCU supporting ZigBee and BLE on CC2650 Development Kit (currently, we use the BLE radio on Raspberry Pi since the Contiki has not yet implemented the BLE stack in its master branch). The integrated emulator (XDS100v3) on the CC2650 Development Kit enables the communication between the Raspberry Pi and the CC2650 MCU through UART. To power the device, we use a USB battery to which a Monsoon power meter [62] is connected to measure the power consumption.

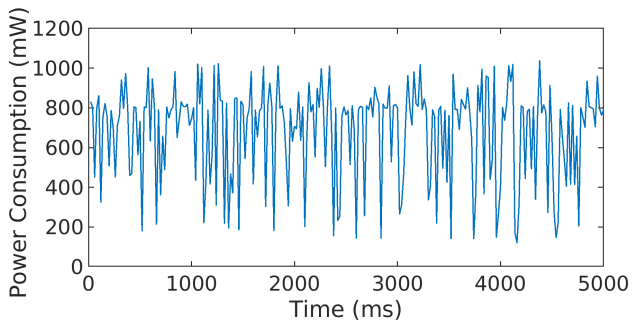
We have realized ARTPoS in Raspbian Linux [63], a Debian based Linux system for Raspberry Pi, and Contiki [64], an operating system for low-power wireless IoT devices. To support WiFi, our ARTPoS implementation adopts the 802.11 MAC and physical layer implementations provided by the Linux kernel and employs the libpcap library for sending and receiving packets to/from the MAC layer. Similarly, our implementation adopts the Linux's BLE implementations and HCI tools to support BLE and uses the 802.15.4 physical layer implementations in Contiki to support ZigBee. Our implementation also adopts the existing UART implementations in Raspbian and Contiki to support the communication between Raspberry Pi and CC2650. In Fig. 3.1, the existing implementations in Raspbian Linux and Contiki adopted by ARTPoS are marked with dash lines, while our new designs are marked with solid lines. WiFi controller, BLE controller, and ZigBee controller are three radio controllers that control WiFi, BLE, and ZigBee radios, respectively. We intentionally implement all modules except the ZigBee Controller in Raspbian Linux, since Raspberry Pi has richer hardware resources. The design of the major modules in ARTPoS are discussed next.

3.2.1 Modeling Engine

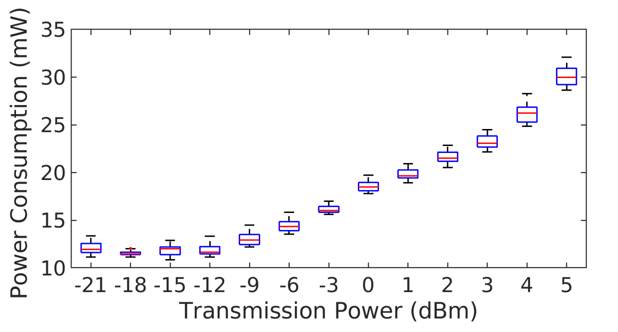
The Modeling Engine generates the power consumption model and link reliability model to support runtime radio and transmission power selection. Most existing solutions for transmission power control for a single radio use a simple



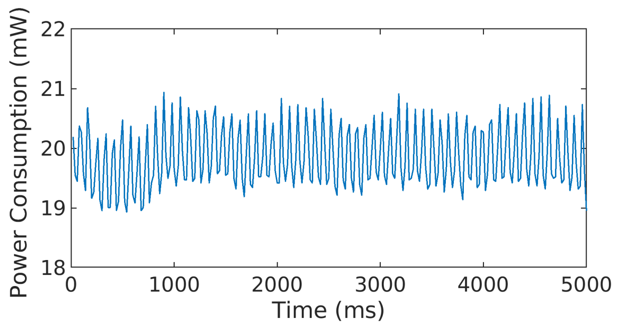
(a) Boxplot of WiFi at 1 dBm to 21 dBm.



(b) A 5-second trace of WiFi at 1 dBm.



(c) Boxplot of ZigBee at -21 dBm to 5 dBm.



(d) A 5-second trace of ZigBee at 1 dBm.

Figure 3.2: Radio power consumptions when WiFi and ZigBee turn on respectively and transmit at the maximum speed. The traces are measured by a Monsoon power meter [62]. In boxplot, central red mark in box indicates median; bottom and top of box represent the 25th percentile (q_1) and 75th percentile (q_2); crosses indicate outliers ($x > q_2 + 1.5 \cdot (q_2 - q_1)$ or $x < q_1 - 1.5 \cdot (q_2 - q_1)$); whiskers indicate range excluding outliers.

power model assuming that using a lower transmission power level leads to lower power consumption. However, this simple model no longer works for a device with multiple radios since the power consumptions have to be compared between different radios. Hence, our Modeling Engine is designed to take real power consumption traces as input and generate power models accordingly. As an example, Fig. 3.2 shows the radio power consumptions when the WiFi and ZigBee radios on our embedded platform turn on respectively and transmit at the maximum speeds at all available transmission power settings. As shown in Fig. 3.2(a), the median power consumption increases from $789mW$ to $905mW$ to $1269mW$ when WiFi is on and the transmission power increases from $1dBm$ to $19dBm$ to $21dBm$, while the median power consumption increases from $11.9mW$ to $18.5mW$ to $30mW$ when ZigBee is on and the transmission power increases from -21dBm to $0dBm$ to $5dBm$ as shown in Fig. 3.2(c). Large variances can be seen in the boxplot in Fig. 3.2(b) and Fig. 3.2(d), which show the 5-second power measurements when WiFi and ZigBee transmit at $1dBm$, respectively. The large variance is caused by the power consumption differences when the radio hardware is at different states, making the first statistical moments (e.g., mean or median) unsuitable to estimate the radio power consumption.

The Modeling Engine also generates the link reliability model based on the PRR measurements at difference distances between the sender and the receiver, and when the sender transmits at different transmission power. PRR can be defined as the fraction of transmitted packets successfully received by the receiver. Our Modeling Engine provides a feature that controls each radio to transmit packets using a single transmission power, then proceeds to the next power in a round robin fashion. With this feature, the PRR measurements for all radios and transmission powers can be done automatically at each distance. However, changing the distance between the sender and receiver has to rely on human operators, introducing labor-intensive measurement overheads. Therefore, it is important to use a frugal set of distance samples that will produce a training data set suitable for effective (subsequent) model development.

Therefore, the **Distance Sample Generator** is designed to generate suitable distance samples based on a feasible communication range and the desired number of distance samples. The desired number of distance samples is decided by the total time allowed for PRR measurements divided by the measurement execution time at each distance. A statistical design of experiments approach, commonly used in Engineering optimization, is employed to generate the distance samples. For instance, the communication range considered, $0 - 200m$ (based on our observed maximum communication range of WiFi/ZigBee/BLE), is divided into three zones. Zone 1, $0 < x \leq 30m$, corresponds to the spatial range in typical home or office-space IoT applications, where a low-power radio like ZigBee is seeing increasing popularity; Zone 2, $30 < x \leq 100m$, corresponds to the spatial range in typical commercial/residential buildings as well as factories and warehouses (i.e., industrial IoT or IIoT applications) where ZigBee becomes progressively less effective, and WiFi is expected to become more dominant; and Zone 3, $x > 100m$, corresponds to the spatial range (typical of emerging cloud robotic and multi-robot applications) where WiFi with greater range capacity will typically dominate. In each of these ranges, we use the *Latin hypercube sampling (LHS)* method to generate 10 distance samples. LHS is a popular approach to generate near-random samples that can provide a relatively uniform coverage of an input space or a probability space [65]. Unlike factorial design or simple Monte Carlo simulations, the size of the sample set yielded by LHS does not scale exponentially with the number of input parameters, thereby making LHS more suitable to design frugal set of experiments (as needed here). A LHS containing n sample points (between 0 and 1) over m dimensions is a matrix of n rows and m columns. Each row corresponds to a sample point. The values of n points in each column are randomly selected, one from each of the intervals, $(0, 1/n), (1/n, 2/n), \ldots, (1 - 1/n, 1)$. We use the optimal LHS implementation, which maximizes the minimum Euclidean distance between the samples [66]. To demonstrate the PRR measurement process, we collect a series of PRR traces by varying the distance between the sender and receiver following the 30 distance samples generated by LHS. Section 3.3.2 will discuss the method that is used to train models of PRR as functions of the respective radio transmission power settings based on our collected PRR traces.

3.2.2 Radio/Transmission Power Selection Engine

The Radio/Transmission Power Selection Engine implements `ARTPoS` core logic. It is designed to facilitate the identification of the best-suited radio(s) and transmission power(s) at runtime. The **Model Container** stores the power consumption model and link reliability model generated by the Modeling Engine. With these two models, the **Optimizer** selects the best radio (or a set of radios) and their optimal transmission power(s) based on the application specified data rate and the throughput of all available links measured by the radio controllers. Section 3.4 will discuss the problem formulation and optimization in detail.

3.2.3 Radio Controllers and User Interface

The Radio Controllers are important design constructs of ARTPoS. Their main purpose is to forward data packets between the application and the radio stacks. The Radio Controllers are responsible for switching on the radio(s) selected by the *Radio/Transmission Power Selection Engine*, keeping the unselected radio(s) off, applying the selected transmission power(s), and routing data packets between the application and the radio stack(s) of the selected radio(s). The **Link Monitor** gathers the runtime link statistics (i.e., throughput and PRR) and feeds them to the Optimizer. To support WiFi, BLE, and ZigBee on our embedded platform, we have implemented three Radio Controllers (i.e., WiFi Controller, BLE Controller, and ZigBee Controller as shown in Fig. 3.1).

The User Interface supports the interactions between our `ARTPoS` and its user. First, it allows the system user to reveal the debugging and operation logs through a SSH connection. Second, it notifies the user to move the device to the next distance when the Modeling Engine finishes the PRR measurements at the current distance. Third, it allows the application to set its desired data rate at runtime.

3.3 Modeling

This section presents the development of tailored regression models with specialized smoothing characteristics, to represent the (uncertain) nodal power consumption and PRR variations as functions of the radio transmission power settings. This modeling approach is aimed to facilitate robust radio and transmission power selection decisions (failure to address these uncertainties undermines radio selection processes, as demonstrated later in Section 3.4.2).

3.3.1 Power Consumption Modeling

The measurements from Section 3.2.1 are used to develop quantitative models of power consumption, as functions of the transmission power setting (p) of the concerned radio. As evident from Fig. 3.2, significant variations, which cannot be solely attributed to change in radio transmission power, are inherent in the measurements. We therefore represent the platform base power consumption with all radios Off ($E_p(V)$), and the respective platform power consumption with only Bluetooth on ($E_b(V, p_b)$), only Zigbee on ($E_z(V, p_z)$), and only WiFi On ($E_w(V, p_w)$) as functions of uncertain parameters V and the respective transmission power of the Bluetooth, ZigBee, and WiFi radios (p_b, p_z, and p_w, respectively).

Here the quantity of interest (QoI), i.e., total power consumption, is a function of the design variable (radio transmission power setting) and a vector of uncertain parameters V, where the latter can be assumed to be outside the control of the designer and not practically measurable in the current context (e.g., radio backOffs caused by failed clear channel assessment and inaccurate power meter reading). Considering the availability of dedicated QoI data (Section 3.2.1), it can be assumed that the uncertainty therein is quantifiable. However, given the observed large variance and non-normal distribution of the platform power consumption data (Fig. 3.2), using the first statistical moments (e.g., mean or median) is deemed not suitable. Secondly, since battery capacity is currently a critical bottleneck in most wireless IoT and embedded system devices, and radios can be a major contributor to power consumption in such devices, we argue that energy over-expenditure (and the uncertainty associated with it) should be perceived as a *risk* – one that can lead to significantly reduced device uptime and/or frequent switching to low performance modes for the concerned device. Hence, we propose to use the notion of **s-risk** [67], to provide a robust or *uncertainty-aware* scalar measure of the risk associated with this expense under any given radio setting.

The notion of s-risk, also known as "conditional-value-at-risk", originated in the Finance domain [68, 69]. Among risk metrics, the s-risk model is well established as a more generalizable model [68] (requires minimal assumptions w.r.t. the underlying process), and thus considered to be a suitable choice in this nascent application setting. We use the example of the platform power consumption with only WiFi On (E_w), to further describe the s-risk concept. Assuming that E_w follows a continuous probability distribution, for a given risk-aversive parameter γ ($0 <= \gamma <= 1$), the s-risk of E_w can be defined as the average value of E_w over its worst $1 - \gamma$ outcomes. Therefore, assuming N samples of E_w are available, s-risk can be expressed as:

$$\begin{aligned} &S_\gamma\left(E_w\left(V, p_w\right)\right) = \frac{1}{\left(1-\gamma\right)N} \sum_{\forall k \in \Gamma} \left[E_w\left(V, p_w\right)_k\right] \\ &\Gamma = \text{set of the highest } (1-\gamma)100\% \text{ values of } E_w \end{aligned} \tag{3.1}$$

It is readily evident from Eq. 3.1 that higher values of γ leads to greater aversion of (energy expenditure) risk or more conservative decisions, in determining the optimal radio settings (optimization approach is described in the next section). From a practical perspective, this "risk-aversive parameter" γ can be designed to be adaptive to the battery state – e.g., the system will use increasingly greater value of γ when the device goes from normal to low and low to critical battery states. Such heuristics could preserve operational feasibility albeit at the cost of reduced data transfer rates. Owing to its ability to consider tails of probability distributions (with the help of higher values of γ) and ease of interpretation and computation, s-risk provides a tractable stochastic measure of the worst-case scenarios. Based on the definition in Eq. 3.1, we compute the following:

- s-risk value of the platform baseline power consumption (S_p) when all radios are Off;
- s-risk value of the platform power consumption with only BLE on (S_b). (Raspberry Pi only supports single transmission power for BLE.);
- s-risk values of the platform power consumption with only ZigBee On (S_z) at the following different transmission power settings: $p_z \in \{-6, -3, 0, 1, 2, 3, 4, 5\}$ dBm;
- s-risk values of the platform power consumption with only WiFi On (S_w) at the following different transmission power settings: $p_w \in \{1, 2, \ldots, 20, 21\}$ dBm.

All s-risk values are computed at a prescribed $\gamma = 0.8$, which here calls for averaging over the worst 50 values in each case.

The s-risk values of the platform power consumption with only WiFi On and only ZigBee On are then separately modeled as linear regressions of their respective transmission settings. A piecewise linear regression is used in the case of WiFi, and a single linear regression is used in the case of ZigBee. The linear regressions provide a smoothing of the large variations in the power traces, while also yielding a monotonically increasing (instead of oscillatory) trend w.r.t. transmission power $-$ which promotes a more robust template for selecting transmission settings (guided by power savings). The trained regression functions can be expressed as:

$$
\begin{aligned}
S_{z,0.8} &= 2.05p_z + 1.89e03, \quad -6 \leq p_z \leq 5 \\
S_{w,0.8} &= \begin{cases} 1.14e01p_w + 2.64e03, & 1 \leq p_w \leq 19 \\ 2.18e02p_w - 1.27e03, & 20 \leq p_w \leq 21 \end{cases}
\end{aligned}
\tag{3.2}
$$

The s-risk models were fitted based on actual power data (collected in the offline experiments). Illustration of the training data and resulting curve fits can be found in [], demonstrating the predictability of the s-risk parameter that captures the energy expenditure risk.

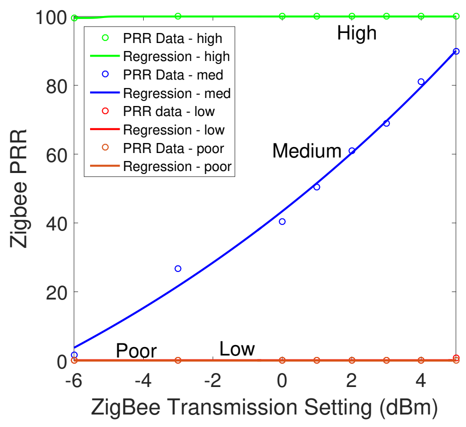
3.3.2 Link Reliability (PRR) Modeling

The PRR measurements from Section 3.2.1 are used to train models of PRR as functions of the respective radio transmission power settings. Here, we particularly develop the PRR models for ZigBee and WiFi, since multiple transmission power settings are available for these two radios on our platform, and they are the ones also considered in the optimal radio and transmission selection process (Section 3.4).

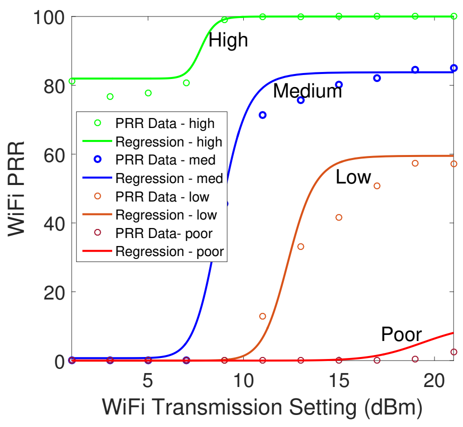
We observe large variations in PRR measurements, especially when the links are in the transitional region. The radio control scheme in practice will usually be unaware of the exact distance between the sender and receiver, as well as of the other uncertain environmental factors affecting the PRR. Instead, what is measurable at runtime are the PRR values being experienced by the individual radios. With this perspective, we propose the state of the system associated with the PRR recordings to be segregated into different performance categories. In this context, the PRR and throughput of an individual radio can also be simultaneously considered, where the categories will then represent the state of the **goodput** (i.e., PRR $\times$ throughput) in that case.

In the current implementation, four categories, namely "*poor*", "*low*", "*medium*", and "*high*" performing states, are defined w.r.t. PRR. For every transmission power setting of a radio (WiFi/ZigBee), the top 25% PRR measurements are assigned to the "*high*" state, the next 25% are assigned to "*medium*" state, the subsequent 25% are assigned to "*low*" state, and the bottom 25% are assigned to the "*poor*" state. Although the recorded (sample) distance between the sender and receiver is not explicitly considered when making this state-category assignments (i.e., all PRR measurements under a given radio setting are pooled together), the assignments are implicitly sensitive to the distance $-$ this is because sender-receiver distance has a strong adverse impact on PRR. The mean of the PRR values categorized under each state for a given transmission setting is then computed to serve as the representative bounding value of the PRR for that state (to be referred to as the *PRR state* or *state-representative PRR* values in the remainder of this chapter). Regression functions are subsequently used to fit the *high*, *medium*, *low*, and *poor* state PRR values of a radio as four separate functions of its transmission settings.

The PRR state values were observed to present S-shaped trends w.r.t. the corresponding radio transmission power settings. This observation led to the choice of logistic regression to model the "PRR-p" relationships between PRR values and transmission power settings. An implementation, called L4P [70], of the four parameter logistic function is used, with the PRR expressed as a function of the



(a) PRR of ZigBee



(b) PRR of WiFi

Figure 3.3: Regression plots of PRR as functions of radio transmission settings; PRR data segregated into *poor*, *low*, *medium*, and *high* states.

radio transmission power, p, as given by

$$PRR(p) = d + (a - d)/(1 + (p/c)^b) \tag{3.3}$$

Here, the four parameters a, b, c, and d respectively represent the minimum asymptote, the stiffness of the curve, the inflection point, and the maximum asymptote. The estimated values of the 8 sets of these four parameters are not listed here, since they are subjective to our recorded PRR measurements, and do not add significant generalized value. Instead, the four logistic functions, that are trained on the *high/ medium/ low/ poor* state PRR values of ZigBee and WiFi, are respectively shown in Figs. 3.3(a) and 3.3(b). It is readily evident from Fig. 3.3 that while capturing the nonlinear S-shaped "PRR-p" relationship, the logistic regression also provides monotonically increasing "PRR-p" functions. Such a positive "PRR-p" correlation is imperative to promoting robust transmission setting modulation – where an optimal scheme should seek to increase the radio transmission power, in response to the need to increase PRR, over the entire range of available transmission power settings.

3.4 Optimization

3.4.1 Problem Formulation

As stated before, the generalized objective of the radio and transmission selection is to adapt to the current needs of the application (under the current environment) in a way that: ***restrict packet loss to within a small/acceptable bound, while platform power consumption attributed to the radios is minimized***. These two criteria, packet loss and power consumption, can be perceived as the *state parameters*; and the choice of the radio type (ZigBee, WiFi, BLE, or any of their combinations) and their transmission power setting can be perceived as *action variables.* This perspective lends to formulating the radio and transmission selection process as an optimization problem, that given the current state of the radio performance chooses the optimum action. The Raspberry Pi only supports single transmission power for BLE; we therefore only consider ZigBee and WiFi in our problem formulation. (We plan to implement our own CC2650 BLE driver under Contiki and include BLE into our optimization as our future work.)

In the remainder of the chapter, the PRR of WiFi and ZigBee, at given transmission settings (p_w and p_z), will be respectively represented by $r_w\,(p_w)$ and $r_z\,(p_z)$ or simply as r_w and r_z, where $0 \leq r_w, r_z \leq 1$; the throughput of WiFi and ZigBee will be expressed in terms of the number of packets transmitted, and represented by h_w and h_z, respectively. The packet size for WiFi and ZigBee is considered to be 64 bytes. The aggregated **goodput** ($G_{w,z}$) of the radios is then given by:

$$G_{w,z}\left(p_w, p_z\right) = h_w r_w\left(p_w\right) + h_z r_z\left(p_z\right) \tag{3.4}$$

If only one of the radios is on, the aggregated gootput reduces to the individual goodput of that radio. The power consumption of the transmitting platform can then be expressed as a function of the data rate (D), the aggregated goodput $G_{w,z}$, the platform baseline power consumption (E_p), and the estimated platform power consumption when radios operate at the given transmission settings (E_w and E_z). The time averaged power consumption of the platform is approximated by:

$$f_E = \min\left(1, D/G_{w,z}\right)\left(E_w + E_z - 2E_p\right) + E_p \tag{3.5}$$

where $(E_w + E_z - 2E_p)$ gives a measure of the power consumption attributable to the active radios. This measure is multiplied by the fraction of the time when the radios need to be active in a given interval; the latter is given by the "*data rate/goodput*" ratio ($\min\left(1, D/G_{w,z}\right)$). When the WiFi is off, $E_w(\mathrm{Off}) = \mathrm{E_p}$ and $r_w(\mathrm{Off}) = 0$; similarly, when the ZigBee is off, $E_z(\mathrm{Off}) = \mathrm{E_p}$ and $r_z(\mathrm{Off}) = 0$. It is also important to note that Eq. 3.5 assumes that the data is split between the two radios based on the ratio of their individual goodputs, and retransmission of lost packets is enabled in the system.

The generalized optimization problem, with the WiFi and ZigBee transmission settings (p_w and p_z, respectively) serving as the decision variables, can therefore be defined as follows:

$$
\begin{aligned}
&\min_{p_w, p_z} \quad f_E\left(p_w, p_z, h_w, h_z\right) \\
&\text{s.t.} \\
&\qquad 1 - \min\left(1, \frac{D}{G_{w,z}\left(p_w, p_z\right)}\right) \geq \epsilon \\
&\text{where} \\
&\qquad p_w \in \{\text{Off}, 1, 2, \ldots, 20, 21\} \\
&\qquad p_z \in \{\text{Off}, -6, -3, 0, 1, 2, 3, 4, 5\}
\end{aligned}
\tag{3.6}
$$

where the tolerance parameter ϵ represents a safety margin in the "*data rate/goodput*" ratio; e.g., $\epsilon = 0.1$ indicates a safety margin of 10% in the "*data rate/goodput*" ratio. It is important to note that both the objective function, f_E (Eq. 3.5), and the "*data rate/goodput*" (Eq. 3.6) constraint are nonlinear, since the PRR is a nonlinear function of the radio transmission power (as seen from Fig. 3.3). In addition, owing to the uncertainties in the PRR and throughput of the radios, and uncertainties in the power consumption of the platform, both the objective and constraint functions are also uncertain. As a result, we have an integer non-linear programming (INLP) problem with uncertainties. Although the INLP problem is NP-hard [71], the relatively limited number of transmission power settings that the two radios can assume (WiFi: 22 and ZigBee: 9) alleviates the computational burden of solving this optimization at runtime. Instead of formulating the optimization under uncertainty as a classical (computationally costly) reliability based optimization problem, uncertainties are addressed apriori using the combination of s-risk measures of power consumption and regression modeling of PRR and s-risk measures (as presented in Section 3.3). The online execution time of solving this optimization problem is presented in Section 3.5.1.

An offline optimization study illustrating the impact of the PRR and power consumption uncertainties (when left untreated) on the radio selection decisions, and the design of our online optimization scheme for runtime radio and transmission selection, are discussed next.

3.4.2 Study on the Impact of Uncertainties

An offline optimization study is set up to investigate how the radio selection is affected by the environmental uncertainties (that cause ill-predictable PRR variations) and systemic uncertainties (that cause power consumption variations). Hence, in this study, we deliberately neither employ any smoothing operation on the empirical data nor use the regression models developed in Section 3.3.

Optimization is performed for different sample combinations of distance between sender and receiver (X) and data rate (D), where $X \in \{10, 20, 30, \ldots, 150\}$ m and $D \in \{25, 50, 75 \ldots, 150\}$ packets/s. A conservative safety margin of 20% ($\epsilon = 0.2$) is imposed on the data rate/goodput ratio. For a given distance, data rate, and radio transmission settings (p_w, p_z), the objective function is evaluated by directly computing the s-risk value of f_E (Eq. 3.5) from the platform power measurements data pertaining to the stated radio transmission settings and the PRR measurements data pertaining to given distance and radio transmission settings (Section 3.2); a risk-aversive parameter of $\beta = 0.8$ is used here. Considering the comparatively smaller variance in the throughput measurements and the focus of this work on dynamic systems (where distance variation mainly affects PRR), the throughput of ZigBee and WiFi is fixed at their respective measured median

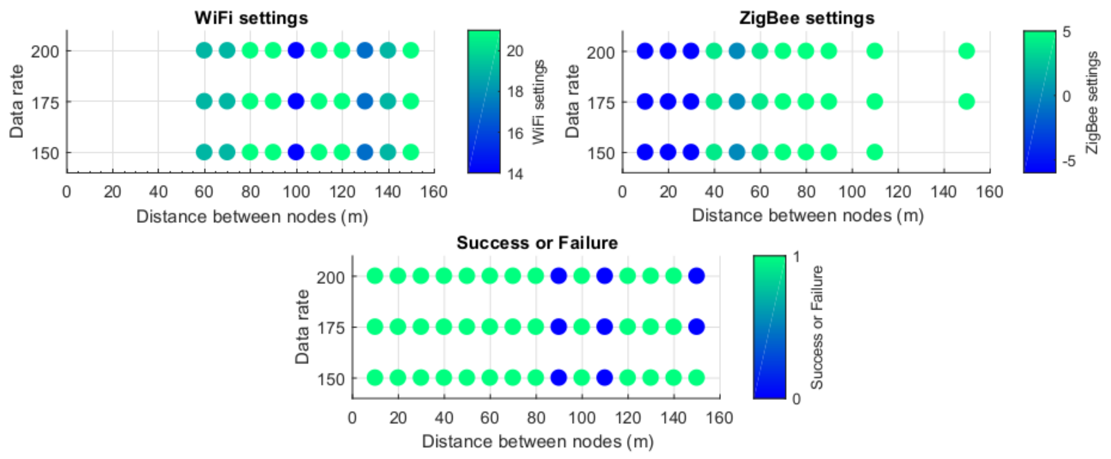


Figure 3.4: Offline study (without smoothing measures or regression models): Top: Optimal transmission power settings of WiFi and Zigbee when operating together; Bottom: success (= 1) or failure (= 0) in meeting the "data-rate/goodput" ratio constraint for different distance and data rate combinations.

values ($h_w = 800$ packets/s and $h_z = 225$ packets/s).

Since only a small set of radio settings are available – i.e., 22×9 possible combinations of (p_w, p_z) – those violating the data rate/goodput ratio constraint are first filtered out; then a simple min-search is employed to identify the optimal feasible setting, p_w^*, p_z^*, that yields the minimum power consumption. This process is performed for all the sample combinations of sender-receiver distance and data rate. The radio transmission setting decisions yielded by this uncertainty-sensitive optimization is shown in Fig. 3.4. For illustration purposes, the results for three data rates (150, 175, and 200 packets/s) are shown. In Fig. 3.4, the X-axis and Y-axis respectively represent the sender-receiver distance and the data rate; in the top two plots, the color of the circles represent the optimal WiFi and ZigBee transmission settings in dBm; and a missing circle indicates that particular radio was set to "OFF" (for the given data rate/distance sample). The last plot in Fig. 3.4 indicates whether the optimal radio setting succeeded (= 1) or failed (= 0) to satisfy the data-rate/goodput ratio constraint (in Eq. 3.6).

The impact of noise/uncertainty of the empirical data (driving the nominal decisions) is apparent in the offline optimization results as shown in Fig. 3.4. For example, it can be seen that when increasing the sender-receiver distance, the radios often switch back and forth between higher and lower settings (instead of a more robust monotonic variation); secondly, no feasible/successful radio setting combination is found for distances of $90m$ and $110m$, although feasible/successful settings were found for higher distances of $120 - 140m$. **These observations highlight the detrimental impact that directly using recorded data (with their associated uncertainties) can have on any empirical decision-making strategy.** This directly motivates 1) the uncertainty-aware power consumption and PRR models developed in Section 3.3, and 2) the design of the two *online* algorithms that use these models to offer robust solutions, which will be described in the next sub-sections.

3.4.3 Fast Online Optimization (*ARTPoS*)

The *ARTPoS* online optimization artifact is developed to serve as a first foray into training a light-weight solution for runtime selection of radio and transmission power under an energy-scarce and uncertain/dynamic environment − typical of application domains such as home/commercial area networks or highly mobile networks. The online scheme should be able to process, interpret, and optimally respond to the uncertainties, without resorting to expensive uncertainty quantification and typical reliability-based optimization techniques. These latter techniques are generally not suited to be executed at runtime on embedded systems with humble computing capacities.

Our approach aims to construct a novel runtime scheme with the following desirable characteristics: (i) **lightweight execution**, (ii) **uncertainty-awareness**, and (iii) promotion of a **power-saving radio/transmission selection policy**. It is important to reiterate that the unique models of power consumption (s-risk models) and PRR (logistic regressions), presented in Section 3.3, are particularly aimed at enabling this light-weight runtime scheme. Drawing parallels to robust control and Markov Decision Processes, the overall objective of the online scheme can be stated as: to maintain/accomplish desirable values of the state parameters (e.g., goodput and platform power consumption) under a dynamic and uncertain environment, by optimally modulating the action variables (i.e., selection of radio(s) and transmission setting(s)).

A look-up table system (*radio-settings-table*) is first generated. Each row (i) and each column (j) of this table respectively corresponds to a WiFi and a ZigBee transmission setting (p_z^j, p_w^i); the table thus comprises a total of 22×9 cells (See Eq. 3.6), where each cell C_{ij} contains one scalar value and two 4-tuples, as shown below:

$$C_{ij} = \{E(p_{z,j}, p_{w,i}),\ R(p_{z,j}), R(p_{w,i})\}$$

$$\begin{aligned}
&E(p_z^j, p_w^i) = S_{z,0.8}\,(p_z^j) + S_{w,0.8}\,(p_w^i) - 2S_{p,0.8} \\
&R(p_{z,j}) = \left(r_{z,j}^{\text{high}}, r_{z,j}^{\text{medium}}, r_{z,j}^{\text{low}}, r_{z,j}^{\text{poor}}\right) \\
&R(p_{w,i}) = \left(r_{w,i}^{\text{high}}, r_{w,i}^{\text{medium}}, r_{w,i}^{\text{low}}, r_{w,i}^{\text{poor}}\right) \\
&\text{where} \quad i = 1, 2, \ldots, 22; \qquad j = 1, 2, \ldots, 9.
\end{aligned} \tag{3.7}$$

In Eq. 3.7, the scalar $E(p_{z,j}, p_{w,i})$ represents the power consumption attributed to the active radios, when operating at the associated transmission setting combination (p_z^j, p_w^i); it is derived from the s-risk measures of power consumption (Section

3.3.1), where the s-risk value of the platform baseline power consumption with both radios Off ($S_{p,0.8}$) is estimated to be $1831mW$; the s-risk values of the platform power consumption with ZigBee on ($S_{z,0.8}$) and that with WiFi on ($S_{w,0.8}$) are estimated from the linear regressions in Eq. 3.2.

The two 4-tuples in Eq. 3.7, $R(p_{z,j})$ and $R(p_{w,i})$, represent the four PRR values corresponding to the *high*, *medium*, *low*, and *poor* operational (or performance) states of ZigBee and WiFi, respectively, at the corresponding transmission settings. These state values are given by the PRR regression functions developed in Section 3.3.2 (Figs. 3.3(a) and 3.3(b)).

It is important to note that in practice, the look-up table is stored/loaded in a more compact form, instead of the 22×9 table (described here for ease of illustration). Since the WiFi and ZigBee settings (i, j) are essentially independent of each other, the look-up table can be stored in the actual test bed in a form that yields a frugal set of "$1 + (5 \times (22 + 9))$" floating point values, making it highly effective for fast runtime decision-making on embedded devices.

The runtime radio and transmission selection algorithm/program, that uses this lookup table, is designed as a four-step process: ***sense***$\rightarrow$***classify***$\rightarrow$***predict***$\rightarrow$***search***. A pseudocode of this runtime program is given in Algorithm 1, and the individual steps are described below.

• ***Sense***: The online process measures PRR (reported by the receiver) and throughput of each radio at a desired sampling frequency; it computes the data rate/goodput ratio (D^t/G^t) based on the time averaged values of PRR and throughput over the last time window t. If the constraint, $1 - D^t/G^t \geq \epsilon$, is violated, it invokes the succeeding steps; otherwise, no change is made. In addition, the

Algorithm 1: ARTPoS

```
Read: x, y, prrw, prrz, Dt, hw, hz; // Input of current state
    variables from receivers, Sense stage
function FindNearest(prrx, x):
    // Classify stage
    if $min(|prrArray - prrx|) > prrLimit$ then
        prrState = min(|prrArray − prrx|);
        refitArray = RefitData(prrArray, x, prrx); // ARTPoS-irp refit
        return min(refitArray) satisfying |refitArray − prrx|;
    else
        return min(prrArray) satisfying |prrArray − prrx|;
    end
end function
function SearchFunc(x, y, prrw, prrz, Dt, hw, hz):
    return Dt/(hw * FindNearest(prrw, x) + hz * FindNearest(prrz, y));
end function
exclude ← powerTable[x+1, y+1]; // Power values to omit in
    search
for (a, b) in powerTable do
    if powerTable[a, b] in exclude then
        continue;
    end
    currentSettings = SearchFunc(a, b, prrw, prrz, Dt, hw, hz);
        // Search stage
    if $currentSettings > 0.9$ then
        continue;
    end
    append (a, b) to feasibleSettings;
    append powerTable[a, b] to feasiblePower;
end
return feasibleSettings[index of min(feasiblePower)]; // Predict stage
```

Algorithm Nomenclature:
prrw, *prrz*: Packet reception ratio of WiFi and ZigBee radios.
Dt: Data rate.
prrArray: Array of pre-generated PRR values that *prrw* and *prrz* are classified against.
refitArray: Array of PRR values including new *prrw* and *prrz* that violates the *prrLimit* threshold (only in ARTPoS-irp).
SearchFunc: function implementing *Search* stage to evaluate the datarate to goodput ratio.
powerTable: pre-computed lookup table from *Predict* stage.
feasibleSettings: array of WiFi and Zigbee dBm values that satisfy search conditions.
feasiblePower: array of power values (from lookup table) that satisfy search conditions.

process computes and checks if the relative change in the D/G ratio is greater than 10%, i.e., $|D^t/G^t - D^{t-1}/G^{t-1}| > 0.1$. If this criteria is met, the succeeding steps are again invoked; otherwise no changes are made. The frequency of the constraint computation and the D/G change computation depends on the designer's preferences. More risk aversive strategies will call for higher frequency of the former, and more energy-conscious strategies will demand higher frequency of the latter. Too frequent changes however may not be recommended, as it might entail unnecessary computing overhead on the system.

• ***Classify***: If the sense process invokes the succeeding steps, first, the current state of each radio's performance, (p^t_w, r^t_w) and (p^t_z, r^t_z), is classified into the *high*, *medium*, *low*, and *poor* (or in-between) state categories. This is accomplished by the following rule: *Classify the current state of the WiFi into lying at one or between the two categories, whose associated PRR values immediately bound the measured PRR.* For example (using Fig. 3.3(b)), if the PRR of WiFi transmitting at $14dBm$ is 70%, then its performance/operation is classified to currently lie between the "*medium*" and "*low*" states; or if the PRR of WiFi transmitting at $4dBm$ is 90%, then its operation is classified into purely "*high*" state. A similar rule applies to ZigBee as well. More sophisticated classification schemes, such as using Bayes rule, can also be readily implemented within this process. This being the first implementation of this novel online scheme, the simpler interval based classification is instead employed here.

• ***Predict***: After the classification step, the D/G constraint (where $G = h^t_w r^t_{w,ij} + h^t_z r^t_{z,ij}$) and the energy objective function (f_E) are evaluated for each cell of the *radio-settings table*, where the latter is given by:

$$f^t_{E,ij} = \min\left(1, \frac{D^t}{h^t_w r^t_{w,i} + h^t_z r^t_{z,j}}\right) E(p^j_z, p^i_w) + S_{p,0.8}$$
$$\text{where} \quad i = 1, 2, \ldots, 22; \quad j = 1, 2, \ldots, 9 \tag{3.8}$$

where the PRR values of ZigBee and WiFi for each cell of the lookup table $(r^t_{w,ij}, r^t_{z,ij})$ correspond to the classified category. More specifically, a linear interpolation is used. Taking the previous example of PRR of WiFi transmitting at $14dBm$ to be 70% – where its operational state is estimated to lie between the "*medium*" and "*low*" categories, the expected PRR of WiFi (at that time point) for say $12dBm$ will be given by:

$$r^t_{w,12} = r^{\text{low}}_{w,12} + \frac{r^t_{w,14} - r^{\text{low}}_{w,14}}{r^{\text{medium}}_{w,14} - r^{\text{low}}_{w,14}}\left(r^{\text{medium}}_{w,12} - r^{\text{low}}_{w,12}\right) \tag{3.9}$$

For purely *high* or purely *poor* states, 100 and 0 are used as the respective upper and lower bounds for the interpolation.

- ***Search***: Once the expected power consumption ($f_{E,ij}$) and the D/G constraint has been computed for all 22×9 ZigBee/WiFi settings, those violating the D/G constraint are first filtered out. A min-search is then executed to identify the optimal ZigBee/WiFi setting, $(i, j)^*$, as the one that yields the smallest value of $f_{E,ij}$. The system immediately switches to this new setting. This step can be expressed as:

$$\begin{aligned} &\min_{i,j} \quad f^t_{E,ij} \\ &\text{subject to} \qquad 1 - \frac{D^t}{h^t_w r^t_{w,ij} + h^t_z r^t_{z,ij}} \geq \epsilon \\ &\text{where} \qquad i = 1, 2, \ldots, 22; \quad j = 1, 2, \ldots, 9 \end{aligned} \tag{3.10}$$

In practice, the filtering of feasible solutions and searching for the optimal solution are both performed in computational efficient ways − e.g., the filtering is initiated by searching from the highest setting, $(p_z^j, p_w^i) = (5, 21)$dBm, and moving somewhat diagonally, until a setting (k, l) is reached where the constraint is violated; all other lower settings (i.e., $\forall(i \leq k, j \leq l)$) are filtered out without computing the constraint.

The median execution time of ARTPoS online optimization is $49ms$ on an ARM processor. Section 3.5.1 will present our micro-benchmark evaluations in detail.

3.4.4 Online Optimization with Insitu Refinement of PRR Models (***ARTPoS-irp***)

ARTPoS-irp is our first step towards a system that is also capable of judging how far the real environment (during operation) deviates from the offline training environment, and adapts its models online in order to provide more reliable decisions. The modified algorithm mainly extends the *Classify* step in the ARTPoS system (see Section 3.4.3) with the aim of increasing the reliability with which the radio's performance ((p_w^t, r_w^t) and (p_z^t, r_z^t)) is classified into the *high, medium, low, poor* state categories. This is achieved by identifying significant deviations (from the offline trends) and responding to it by dynamically refitting the PRR regression models used in the *Classify* step. As the radio's performance now seeks to be reflective of the environment in which the system operates, the algorithm is expected to become more robust in its adaptation.

The refit is invoked by consistent over/underestimation of the PRR state compared to the classified curves (given in Fig. 3.3). In *ARTPoS-irp*, the measure of over/underestimation is through the observation of the difference between the measured PRR value and the PRR value at the given transmission setting (dBm) based on the state it is classified under. If the difference exceeds a certain threshold, $r_{TH} = 0.3$ for n consecutive time steps, the refit is performed; for example, perform refit of the WiFi PRR curve if:

$$|r_{w,i}^{\text{classified}} - r_{w,i}^{\text{meas}}| > r_{TH}, \qquad \text{for } n \text{ consecutive time steps} \tag{3.11}$$

Here, n is set at 5, and $r_{w,i}^{\text{classified}}$ is the PRR given by the curve into which the current state has been classified, and $r_{w,i}^{\text{meas}}$ is the online measured PRR value. If the difference does not exceed the threshold, the refit is not invoked and ARTPoS-irp behaves identically to ARTPoS.

Next, the violating PRR values are added to the existing dataset, and the refit is performed using the logistic regression [70] described in Eq. 3.3. In order to prevent ever-growing size of the dataset during operation, a forgetting strategy can be used (after a threshold size is exceeded) where every time new data is added for refit the oldest data at the corresponding transmission dBm can be removed from the set.

To illustrate the role played by this online updating strategy, we provide a representative example in Fig. 3.5. Here, the leftmost plot (Fig. 3.5(a)) shows the original offline trained PRR models for WiFi. The next plot (Fig. 3.5(b)) shows the PRR models after one round of updating invoked by new deviating data at 7dBm classified under the "*medium*" category PRR model (note that the blue curve fit, corresponding to "Model - med" has got updated). The rightmost plot (Fig. 3.5(c)) shows the PRR models after another round of updating, in this case

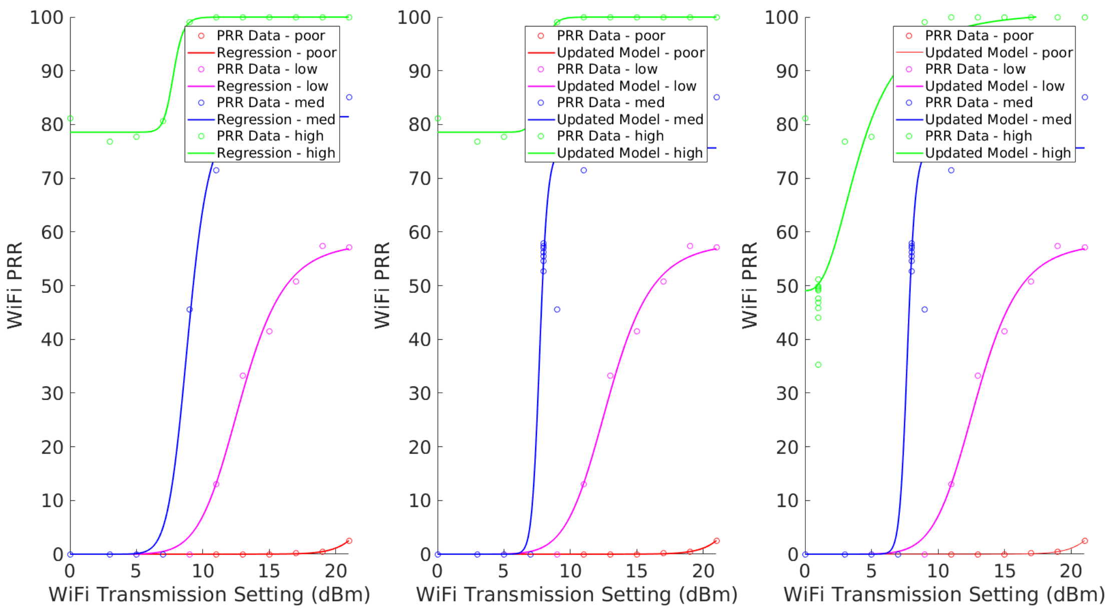


Figure 3.5: WiFi PRR Models: showing how the offline trained model gets updated online in response to new PRR data that deviate from the offline fits

invoked by new deviating data at 1dBm classified under the "*high*" category (note that the green curve fit, corresponding to "Model - high" has got updated). It is important to note from Fig. 3.5 that, the observed effectiveness of adapting the PRR models to the varying runtime environment is attributed to both the new online updating scheme in ARTPoS-irp and the original choice of the (logistic) regression fitting.

3.5 Evaluation

To examine the efficiency of ARTPoS and ARTPoS-irp, we perform a series of experiments on our embedded platform presented in Section 3.2. We first measure the overhead of the key operations such as the time duration of the optimizer selecting the best radio(s) and needed transmission power(s) and the overhead attributed to turning the radio(s) On and Off. We then evaluate ARTPoS/ARTPoS-irp's impact on power consumption and link reliability, and compare their perfor-

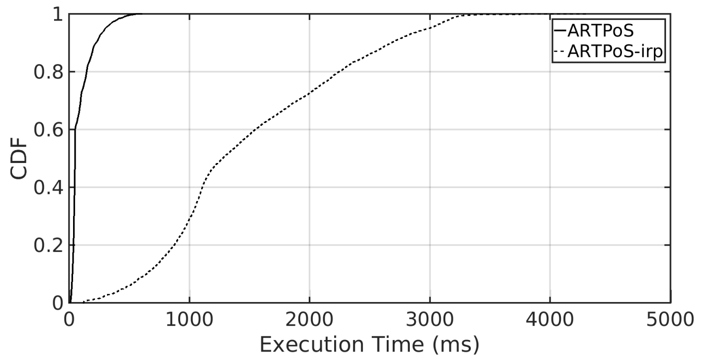


Figure 3.6: CDF of the time duration for ARTPoS and ARTPoS-irp to determine the optimal radio and transmission power.

mance against three baselines. A power meter from Monsoon Solutions [62] is connected to the sender to measure the power consumption.

3.5.1 Micro-Benchmark Experiments

We first evaluate the time duration taken by the two online optimal approaches to select the best radio(s) and minimum needed transmission power(s). We record the time of the events when the input is fed into the optimizer and the output (i.e., radio and transmission power selection) is generated. For this experiment, we repeat the measurement 10,000 times for both ARTPoS and ARTPoS-irp (with refit), using randomly generated inputs, on our 1.2GHz 64-bit quad-core ARMv8 CPU platform. In order to show the difference to ARTPoS, we force ARTPoS-irp to invoke its refit every time by feeding in randomized inputs, since ARTPoS-irp's behaviour is identical to ARTPoS's without invoking its refit. The difference of the execution time between the two methods represents the time taken to perform the refit triggered by estimation errors. Figure 3.6 compares the cumulative probability density (CDF) of the algorithm execution time of ARTPoS and ARTPoS-irp. As shown in Fig. 3.6, the median execution time of ARTPoS is $49ms$ (consuming $13.5mJ$ more energy than CPU idling), where 90% and 99% of the experimental

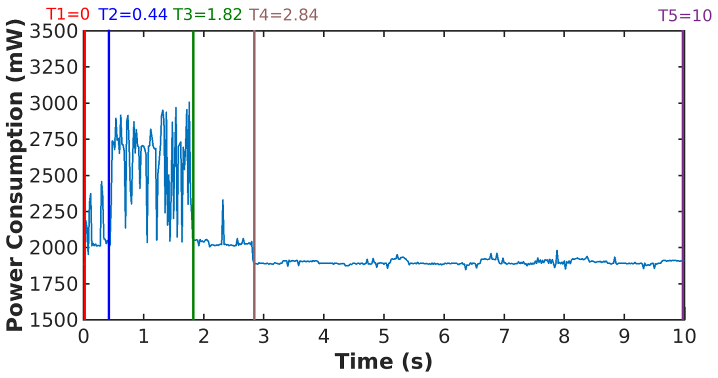
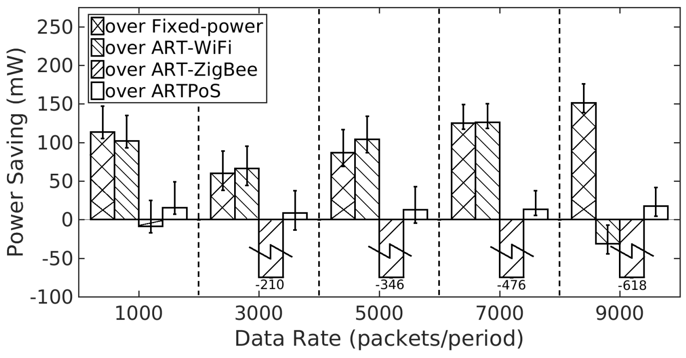


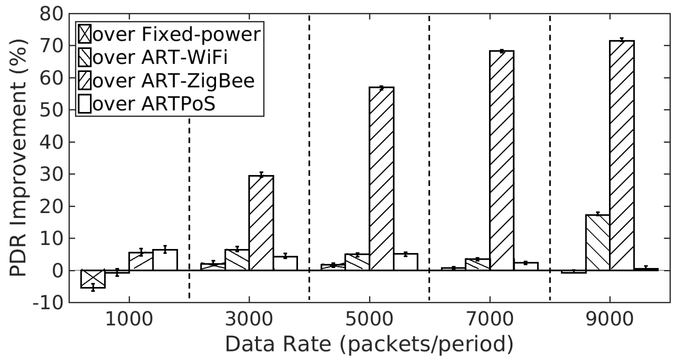
Figure 3.7: Radio activities when the WiFi controller manages packet transmission in a 10s period; averaged power consumption over the first three time periods, $T_1 \to T_2$, $T_2 \to T_3$ and $T_3 \to T_4$, respectively are $2.09mW$, $2.61mW$ and $2.03mW$.

runs finish within less than $225ms$ and $456ms$, respectively. In comparison, the median execution time of ARTPoS-irp is $1273ms$, where 90% of the experimental runs finish within $2675ms$; this additional computing burden can be directly attributed to the refitting of the PRR function (via logistic regression) performed insitu in ARTPoS-irp. This burden can be alleviated by increasing the deviation threshold and/or the number of consecutive time steps for which deviation is allowed (refer Eq. 3.11) before invoking the refit; future work would explore how computational efficiency trades-off with energy and link reliability performance in this context.

We also measure the time duration and energy consumption of other key operations in ARTPoS and ARTPoS-irp. Figure 3.7 shows an example power consumption trace where the WiFi controller switches On the WiFi radio, transmits 1000 packets, and then switches Off the radio. The platform takes $T_2 - T_1 = 0.44s$ and consumes $0.92J$ of energy to turn On the radio and set its transmission power. Transmitting 1000 packets takes $T_3 - T_2 = 1.38s$, while turning Off the radios takes $T_4 - T_3 = 1.02s$. The platform consumes $3.60J$ and $2.07J$ of energy to transmit the data and turn Off the radio, respectively. The radios are kept Off for the rest



(a) Power consumption saving provided by ARTPoS-irp over the baselines and ARTPoS



(b) PDR improvement provided by ARTPoS-irp over the baselines and ARTPoS

Figure 3.8: Power consumption and PDR differences between our approaches (ARTPoS and ARTPoS-irp) and the baselines (Fixed-power, ART-WiFi and ART-ZigBee) at different data rates.

of the period $T_5 - T_4 = 7.16s$. These results demonstrate the efficiency of the optimizer and the radio controllers, as well as the advantage of turning the radios Off after transmissions in each period, and also illustrate the significant need of developing new low-power platforms for IoT applications to achieve lower baseline power consumption.

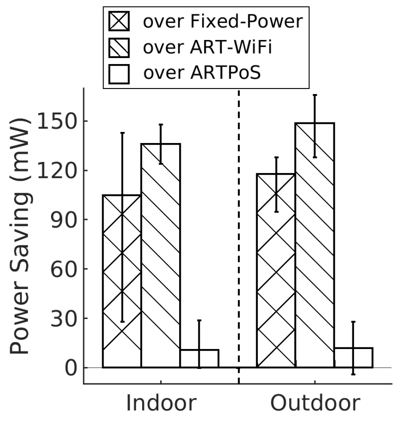
3.5.2 Impact on Power Consumption and Link Reliability

To understand how the proposed methods impact power consumption and link reliability, we performed a set of experiments comparing the performance of ARTPoS and ARTPoS-irp with three baselines. In all experiments, we deploy a benchmark application on top of the ARTPoS and ARTPoS-irp by generating data packets periodically. ARTPoS and ARTPoS-irp are configured to perform the radio and transmission power selection in each period (i.e., $10s$) based on the measured PRR and throughput of the ZigBee and WiFi links. If the then-active radio and transmission power setting is found to be the best-suited, it is retained; else the ARTPoS/ARTPoS-irp switches to a new best-suited setting. Non-overlapping channels are used for ZigBee and WiFi to avoid interference. Radios are turned Off after the last transmission in each period and the unselected one is kept Off to reduce power consumption for our approaches and the baselines. If both radios are selected for use, packets are partitioned based on their throughput ratio, allowing the platform to sleep earlier and save energy. Due to the lack of a baseline that jointly optimizes the selection of both radio and transmission power, we extend the ART [14], a practical state-of-the-art transmission power control approach designed for ZigBee, and create three baselines: one with only ZigBee radio on running ART (*ART-ZigBee*), one with only WiFi radio on running ART (*ART-WiFi*), and one with both radios on operating at their default powers, i.e., 21dBm for WiFi and 5dBm for ZigBee (*Fixed-power*).

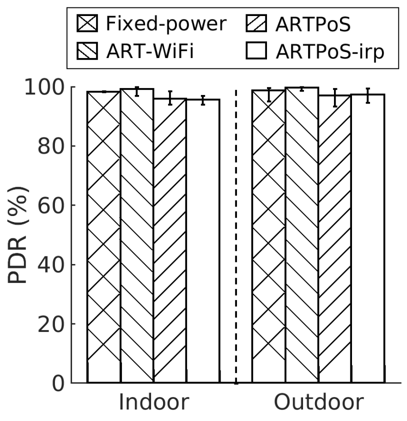
We performed five experimental runs, respectively with Fixed-power, ART-WiFi, ART-ZigBee, ARTPoS, and ARTPoS-irp, in a round robin fashion to minimize the temporal effects of the dynamic wireless environment (for fair comparison). Figure 3.8 shows the power consumption and packet delivery rate (PDR) comparisons between our approaches and the baselines. To explore ARTPoS-irp's performance under different traffic demands, we repeated the experiments by controlling the application to generate data at different rates. Under each data rate and approach, we repeat the experiments five times and present the confidence intervals in Fig. 3.8.

As shown in Fig. 3.8(a), both of our proposed methods, ARTPoS and ARTPoS-irp, provide significant power savings compared to the Fixed-power and ART-WiFi baselines. For example, our ARTPoS-irp reduces the average power consumption by $114mW$ and $102mW$ over Fixed-power and ART-WiFi, respectively, when the data rate is 1000 *packets/period*. Similarly, ARTPoS-irp achieves significant power savings over Fixed-power and ART-WiFi at higher data rates ($60.1mW$ and $66.2mW$ at 3000 *packets/period*, $86.5mW$ and $104mW$ at 5000 *packets/period*, and $125mW$ and $126mW$ at 7000 *packets/period*). As a comparison for power saving values, the CC2650 radio consumes $30mW$ power when transmitting at $5dBm$ [58]. The original ARTPoS demonstrates significant improvements over the baselines (Fig. 3.8). It is however important to note that ARTPoS-irp does outperform ARTPoS, by providing 0.4% to 6.4% greater PDR (3.7% increase on average), while consuming $8.7mW$ to $17.3mW$ less power ($13.5mW$ decrease on average) than ARTPoS for each data rate. These observations provide direct evidence for the conceived benefits of the insitu (PRR model) refinement incorporated in ARTPoS-irp.

Compared to ART-ZigBee, ARTPoS-irp consumes $8.5mW$ more power at the lowest data rate since it initially turns on the WiFi and ZigBee radios to measure their channel conditions. More importantly, although ARTPoS-irp consumes more power than ART-ZigBee, the latter is not able to deliver satisfactory PDRs at high data rates because of the ZigBee's limited bandwidth (i.e., the average PDRs under ART-ZigBee are 68.7%, 44.6%, 31.0%, and 25.0% when the data rate is



(a) Power consumption saving provided by ARTPoS-irp over the baselines and ARTPoS



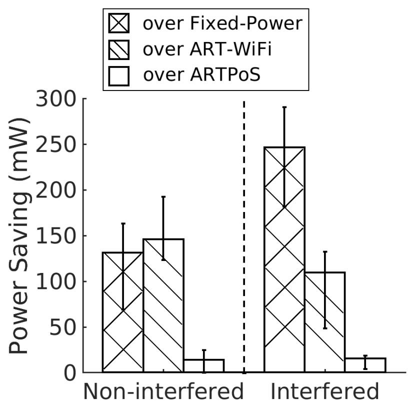
(b) PDR achieved by the baselines and our approaches

Figure 3.9: Power consumption and PDR comparison between our approaches (ARTPoS and ARTPoS-irp) and the baselines (Fixed-power, ART-WiFi and ART-ZigBee) at different locations.

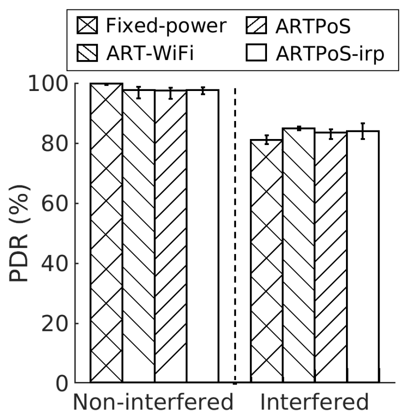
3000, 5000, 7000, and 9000 *packets/period*, respectively, i.e., significantly inferior to ARTPoS-irp and ARTPoS (as seen from Fig. 3.8(b)). Neither WiFi nor ZigBee alone can support the data rate of 9000 *packets/period*, while our ARTPoS-irp and ARTPoS provide satisfactory PDRs by bundling the WiFi and ZigBee radios.

In order to examine ARTPoS-irp's performance under different environments, we set the data rate to 7000 *packets/period* and performed a set of experiments comparing the performance of ARTPoS-irp with ARTPoS and two baselines (Fixed-power and ART-WiFi)[2] at different indoor and outdoor locations. The transmitters and receivers are placed at different rooms in an indoor office environment and in an outdoor open space. Fig. 3.9 shows the power consumption and PDR comparisons between our approaches and the baselines. At each location, we repeat the experiments with each approach five times and present the confidence intervals in Fig. 3.9. As shown in Fig. 3.9(a), both of our proposed methods, ARTPoS and ARTPoS-irp, provide significant power savings compared

[2]We did not run experiments to evaluate ART-ZigBee because the data rate (7000 *packets/period*) is beyond ZigBee's capacity.



(a) Power consumption saving provided by ARTPoS-irp over the baselines and ARTPoS

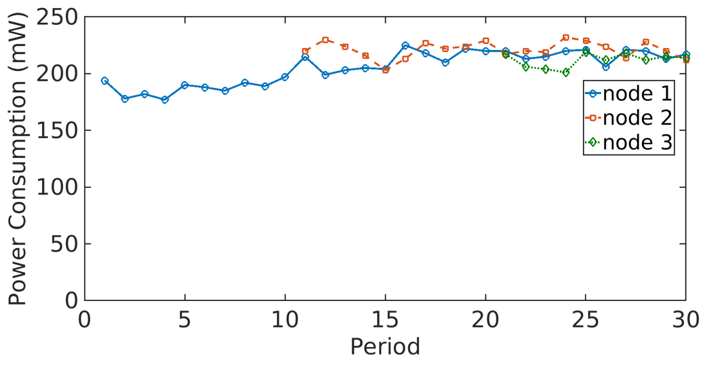


(b) PDR achieved by the baselines and our approaches

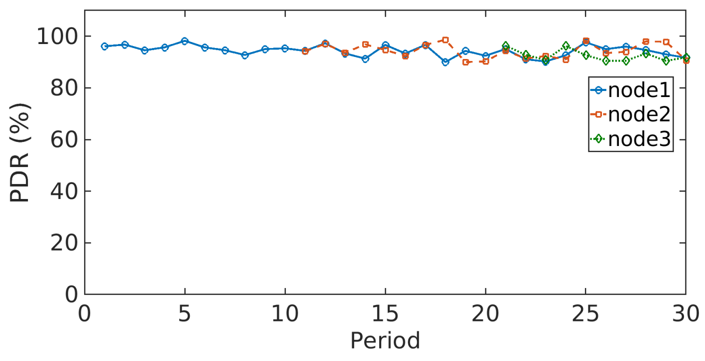
Figure 3.10: Power consumption and PDR comparison between our approaches (ARTPoS and ARTPoS-irp) and the baselines (Fixed-power and ART-WiFi) with and without interference.

to Fixed-power and ART-WiFi baselines. For example, ARTPoS-irp reduces the average power consumption by $105mW$ and $136mW$ over Fixed-power and ART-WiFi, respectively, when performed indoors, and saves $118mW$ and $148mW$ when performed outdoors. As shown in Fig. 3.9(b), ARTPoS and ARTPoS-irp achieve average PDRs over 95% at both indoor and outdoor locations, which are very close to that of Fixed-power and ART-WiFi.

To evaluate ARTPoS-irp's performance under different interference conditions, we set the data rate to 7000 *packets/period* and performed a set of experiments comparing the performance of ARTPoS-irp with ARTPoS and two baselines (Fixed-power and ART-WiFi) with and without interference. We run JamLab [72] on a TelosB mote [73] to generate controlled interference. The jammer is placed one meter away from the receiver. Fig. 3.10 shows the power consumption and PDR comparisons between our approaches and the baselines. Under each channel condition, we repeat the experiments with each approach five times and present the confidence intervals in Fig. 3.10. As shown in Fig. 3.10(a), ARTPoS



(a) Radio power consumption trace



(b) PDR trace

Figure 3.11: Radio power consumption and PDR traces of three transmitters for 30 periods.

and ARTPoS-irp provide significant power savings compared to Fixed-power and ART-WiFi baselines. For example, ARTPoS-irp reduces the average power consumption by 131mW and 146mW over Fixed-power and ART-WiFi, respectively, when performed without interference, and saves 246mW and 109mW with interference. It is notable that more energy is consumed by the WiFi radio when the WiFi channel is interfered, especially for ART-WiFi which uses the WiFi radio only. As shown in Fig. 3.10(b), the average PDRs of all approaches are over 95% without interference and decrease to the range between 81% and 85% with interference.

Finally, we examine the performance of ARTPoS-irp when new nodes join the network. When multiple senders transmit data to a single receiver, the nework is configured to run a TDMA-based MAC to avoid packet collisions. In the experiment, we configure three senders to join the network one by one with the data rate of 3000 $packets/period$. Specifically, only node 1 sends data to the receiver during the first 10 periods (200s). Node 2 begins to transmit at the 11th period, while node 3 joins the network at the 21st period. Fig. 3.11 plots the power consumption and PDR during the experiment. Each node runs ARTPoS-irp to select its radios and transmission powers. As Fig. 3.11(a) shows, each node consumes slightly more power when multiple nodes are present because of the idle listening. For example, the median power consumption of node 1 during the first 10 periods is $188mW$. It increases to $212mW$ during the next 20 periods when two senders transmit. It further increases to $218mW$ when three senders are present. As Fig. 3.11(b) shows, the PDRs remain stable when new nodes join the network, demonstrating the effectiveness of ARTPoS-irp on preserving the link reliability through running a TDMA-based MAC protocol.

The overall experimental results thus show that ARTPoS-irp and ARTPoS can effectively reduce the energy consumption while maintaining satisfactory link reliability, to meet varying network traffic demands under different real/uncertain environments. Moreover, the new ARTPoS-irp consistently delivers superior performance compared to the original ARTPoS, particularly in terms of power consumption, thereby demonstrating the advantage of the novel online adaptation mechanism built into ARTPoS-irp.

3.6 Conclusion and Future Work

Given the dynamic nature of communication in IoT (e.g., moving IoT/robotic units in uncertain commercial/residential/industrial environments), a traditional one-radio-fits-all approach cannot meet the challenges under typically varying operating conditions and traffic. This chapter presents the new ARTPoS system that makes available multiple wireless technologies at runtime and selects the radio(s) and their transmission power(s) most suitable for the current conditions. The selection process aims to preserve link reliability within acceptable thresholds, while minimizing the power consumption of the node attributed to radio operation. To this end, empirical approaches to modeling power and PRR are presented, which allow the system to proactively adapt to large variations in power consumption and link reliability observed runtime. This is followed by the development of two computationally light-weight online optimization schemes, based on a unique sense-classify-predict-search process, with the latter scheme also employing an in-situ (runtime) refinement of the PRR models for added robustness in meeting the QoS objectives. Experimental evaluations of the thus formulated online optimization schemes, and their comparison with different baselines, show that ARTPoS can remarkably reduce the power consumption, while maintaining satisfactory link reliability. We plan to integrate ARTPoS with the low power listening technique to support efficient duty cycling and enable model updating at runtime as our future works. In addition, we are also currently investigating approaches to extend this fundamental radio/transmission selection technique from a one-to-one communication to a many-to-many/network-scale communication framework involving gateways. Decomposed problem formulations and decentralized decision-making are expected to serve as two other core elements in facilitating this important next step in this research.

4 Radio Selection and Data Partitioning for Energy-Efficient Wireless Data Transfer in Real-Time IoT Applications

4.1 Introduction

The importance of real-time wireless data transfer is rapidly increasing for the Internet of Things (IoT) applications. For example, smart glasses worn by a doctor need to transmit real-time data to a hospital information system, which performs face detection and recognition, for real-time interaction with recognized patients within a certain deadline, which is ideally a few hundred milliseconds [74]. As another example, periodic sensor readings from unmanned aerial vehicles (UAVs) should be delivered every second to a georeferencing system that analyzes the data to determine the real-time position and altitude of UAVs [75]. Other emerging IoT applications, e.g., structural health monitoring [76], clinical monitoring [77], and industrial process automation [78, 79], also require real-time wireless data transfer. In such applications, missing data delivery deadlines may result in cognitive distraction, injury, structural damage, or safety hazard. However, it is very challenging to support stringent timing constraints through wireless medium due to its inherent unreliability and timing-unpredictability. Moreover, the energy constraints significantly amplify the challenge, since most of those IoT devices are battery-powered and achieving high energy efficiency is critical for those applications.

Fortunately, embedded system hardware and radio technologies are advancing fast in recent years. As a result, more and more embedded devices are equipped with heterogeneous radios. For example, Firestorm [57] supports ZigBee and Bluetooth Low Energy (BLE) in one device and TI CC2650 [58] integrates those two radios on a single chip. IOT-Gate-iMX7 [60] is an industrial IoT gateway, which supports 4G/LTE, WiFi, Bluetooth, and Zigbee. LX Cellular Core [80] is a small-sized IoT platform, which features 2G/3G, WiFi, BLE, ANT+, LoRa, Taggle, and SigFox. Heterogeneous radios are becoming increasingly available in modern embedded devices, offering new opportunities to use multiple wireless technologies for real-time applications. However, using multiple heterogeneous radios may enhance the timeliness at the expense of higher energy consumption or vice versa. It is even more challenging to strike a good balance between the two potentially conflicting requirements.

This chapter aims to address the previously stated challenges and presents an energy-efficient radio switching and bundling solution to minimize the energy consumption of battery-powered IoT devices for real-time applications and reduce the deadline miss ratio when facing tight deadlines, leveraging the aforementioned hardware advancements. To assure the timeliness, we target at a single-hop application scenario, since most existing solutions relying on multi-hop mesh networks suffer from long latency and high complexity. Our approach conforms to the advanced wireless network technology trend as the industry is investing heavily in network infrastructure to support IoT visions such as smart cities. As a result, more and more access points and edge servers are becoming readily available to support various IoT applications. Specifically, we make the following contributions:

- We formulate the runtime radio switching and bundling as an Integer Linear Programming (ILP) problem;
- We design the *Real-Time radio Selection (RT-Select)* algorithm that optimally and quickly selects between two radios and partitions data between them at runtime to minimize the energy consumption;
- Based on RT-Select, we design the *RT-Select-General* algorithm for the platforms with more radios.
- We design the *Real-Time traffic Balance (RT-Balance)* algorithm that balances the traffic assigned to different radios at runtime to reduce deadline miss ratio when facing tight deadlines.
- We develop the *Real-time Radio Switching and Bundling (RRaSB)* system that runs on our embedded platform equipped with five heterogeneous radios, selectively makes a subset of radios available at runtime, and allows dynamic radio switching and bundling among them;
- We implement RT-Select, RT-Select-General, and RT-Balance in RRaSB and evaluate them experimentally; experimental results show that our RT-Select and RT-Select-General significantly outperform the baseline (GreenBag) and RT-Balance effectively help RT-Select and RT-Select-General reduce deadline miss ratios.

The remainder of the chapter is organized as follows. Section 4.2 introduces our problem formulation. Section 4.3 presents the design of RT-Select, RT-Select-General, and RT-Balance. Section 4.4 describes RRaSB. Section 4.5 presents our experimental evaluation. Section 4.6 concludes the chapter.

4.2 Problem Formulation

In this section, we formulate the runtime radio selection and data partitioning for real-time applications as an ILP problem. We first introduce some related radio characteristics and then define the objective function and constraints of the ILP problem.

We assume that m radios, $R_1, ..., R_m$, are available on an IoT end device. The characteristics of each radio $R_i (1 \leq i \leq m)$ are separated into two categories:

1. variable characteristics related to the bandwidth and reliability of the wireless link between R_i and the IoT gateway:

    - throughput, TH_i, is the maximum number of data packets which R_i is able to successfully deliver to the IoT gateway per second;

    - expected transmission count, ETX_i, is the average number of transmission(s) which R_i needs to attempt to successfully deliver a packet to the IoT gateway.

2. constant characteristics related to energy and time consumption of R_i:

    - switching energy, E_{sw_i}, is the total energy consumed to switch R_i on and off[1];

[1] R_i is turned off by default after it transmits all assigned packets if the future traffic demand is unknown.

- switching time, T_{sw_i}, is the time taken to switch R_i on[2];
- radio base power, P_{rb_i}, is the base power consumed by R_i when the radio is on and idle;
- per-transmission energy, E_{ta_i}, stands for the additional energy consumed by R_i for each packet transmission attempt.

We define the deadline miss ratio as the number of data transfers which are not completed before their deadlines divided by the total number of data transfers. Since the deadline miss ratio directly reflects the performance of real-time applications, we minimize the deadline miss ratio instead of the absolute latency. Thus, our optimization goal is to minimize the radio energy consumption, while meeting the data rate and deadline requirements. To achieve the objective, we select the radio(s) and assign data packets to them. We assume that there are N packets required to be delivered by deadline D. Let us also assume that X_i packets are assigned to radio R_i, where $0 < X_i \leq N$ if R_i is selected or $X_i = 0$ if R_i is not selected. The objective function to minimize is the sender's energy consumption E, which is the sum of the radio switching energy, radio base energy, and radio transmission energy consumed by the selected radios as shown in Eq. 4.1, where the radio base energy is P_{rb_i} multiplied by the transmission time (X_i/TH_i), the radio transmission energy is E_{ta_i} multiplied by ETX_i and X_i, and the set S is composed of the indices of all selected radios:

$$\boldsymbol{min} \left\{ \sum_{i \in S} \left(E_{sw\_i} + P_{rb\_i} \times \frac{X_i}{TH_i} + E_{ta\_i} \times ETX_i \times X_i \right) \right\} \tag{4.1}$$

[2]The time taken to switch R_i off is not included since the radio can be turned off after the deadline if it is not selected for use in the next period.

There are three constraints on variable X_i (the number of packets assigned to R_i): (i) X_i is a non-negative integer not greater than N as specified in Eq. 4.2 (ii) X_i should not exceed the maximum packet delivery capacity of the radio link (X_{max_i}) for the deadline D as stated in Eq. 4.3 and (iii) the total number of packets assigned to all radios should be equal to N as specified in Eq. 4.4. Therefore, the following constraints should be met to satisfy the traffic demand and deadline requirements:

$$0 \leq X_i \leq N \ (X_i \in \mathbb{N}) \tag{4.2}$$

$$X_i \leq X_{max\_i} \equiv (D - T_{sw\_i}) \times TH_i \tag{4.3}$$

$$\sum_{i=1}^{m} X_i = N \tag{4.4}$$

In addition, let us introduce a Boolean variable, Y_i, to indicate whether or not the radio R_i is selected. $Y_i = 1$ if R_i is selected ($X_i > 0$) and $Y_i = 0$ if R_i is not selected ($X_i = 0$).

Given Eq. 4.2–4.4, we simplify the objective function E in terms of variables X_i and Y_i as well as coefficients A_i and B_i as follows:

$$\boldsymbol{min} \left(\sum_{i=1}^{m} [A_i Y_i + B_i X_i] \right) \tag{4.5}$$

where

$$\begin{aligned} A_i &= E_{sw\_i} \\ B_i &= \frac{P_{rb\_i}}{TH_i} + E_{ta\_i} \times ETX_i \end{aligned} \tag{4.6}$$

Eq. 4.2–4.6 form an ILP problem, which is NP-hard.

Many resource-constrained IoT devices cannot afford to execute an ILP solver to solve the problem at runtime for real-time applications. This motivates us to develop lightweight algorithms tailored for the runtime radio selection and data partitioning problem.

4.3 Algorithm Design

Algorithm 2: RT-Select

```
    Input  : $N, D, RC_1, RC_2$
    Output: $X_1, X_2$
1   Compute $A_i, B_i, X_{max\_i} | i = 1, 2$ ;
2   $(idx\_1, idx\_2) = \boldsymbol{sort}\{A_i + B_i \times N \mid i = 1, 2\}$ ;
3   $(idx\_1', idx\_2') = \boldsymbol{sort}\{B_1, B_2\}$ ;
4   if $X_{max\_(idx\_1)} \geq N$ then
5       $X_{idx\_1} \leftarrow N$ ;
6   else if $X_{max\_(idx\_1)} < N$ and $X_{max\_(idx\_2)} < N$ then
7       $X_{idx\_1'} \leftarrow X_{max\_(idx\_1')}$ ;
8       if !Conflict() then
9           $X_{idx\_2'} \leftarrow N - X_{idx\_1'}$ ;
10      end
11  else
12      if $B_{idx\_2} < B_{idx\_1}$ or $A_{idx\_1'}/(B_{idx\_2} - B_{idx\_1'}) > X_{max\_(idx\_1')}$ or
          Conflict() then
13          $X_{idx\_2} \leftarrow N$ ;
14      else
15          $X_{idx\_1'} \leftarrow X_{max\_(idx\_1')}$ ;
16          $X_{idx\_2} \leftarrow N - X_{idx\_1'}$ ;
17      end
18  end
```

One of the primary design goals of our algorithms is to be time-efficient. With the consideration of the demand of fast responses, our decision-making strategies can be processed fast by the IoT devices to guide the runtime radio selection and data partitioning in response to the current wireless link state and application timing requirement. Specifically, we first design the *RT-Select* algorithm that optimally solves the two-radio case of the problem and prove its optimality. Then, based on the insights from the design of RT-Select, we design the

RT-Select-General algorithm to solve the general form of the problem involving m radios. Finally, we design the *RT-Balance* algorithm that balances the traffic assigned to different radios at runtime to reduce deadline miss ratio when facing tight deadlines. All of our algorithms take the inputs of the traffic demand (i.e., N packets) and the delivery deadline D specified by the application and the pre-measured radio characteristics. While RT-Select and RT-Select-General output the radio selection decision, RT-Balance adjusts the traffic assignments at runtime and outputs the result whether the deadline is met successfully. For simplicity, we use RC_i to represent the characteristics of each radio R_i including $TH_i, ETX_i, E_{sw\_i}, T_{sw\_i}, P_{rb\_i}$ and E_{ta_i} (see Section 4.2).

Please note that an embedded device may not allow to use some of its radios simultaneously due to hardware conflicts. For example, the ZigBee and BLE radios on the TI CC2650 [58] cannot operate simultaneously, since they share a single DSP modem and a digital PLL. Our algorithms always consider such hardware conflicts when selecting radios.

4.3.1 RT-Select Algorithm for Selection between Two Radios

Algorithm 2 shows RT-Select algorithm that selects between two radios to minimize the energy consumption, while meeting the application specified traffic demand and deadline requirements. We have proven the optimality of Algorithm 2 [3]. RT-Select first computes the A_i, B_i, and X_{max_i} values for both radios based on Eq. 4.6 and Eq. 4.3 (Line 1). It then sorts the two radios based on the energy consumption for each radio to transmit N packets by itself ($A_i + B_i \times N$) and stores the radio indices to (idx_1, idx_2) in ascending order (Line 2). Therefore, the radio R_{idx_1} is more energy-efficient than R_{idx_2}. Similarly, RT-Select sorts the two radios based on the average energy consumption per packet B_i without considering radio switching energy consumption A_i and stores the radio indices to (idx_1', idx_2') in ascending order (Line 3). Therefore, the radio $R_{idx\_1'}$ is more energy-efficient than $R_{idx\_2'}$ without considering radio switching energy consumption A_i. The radio hardware conflict checker "*Conflict*()" gets the boolean information on whether there is a hardware conflict between the two radios which prevents them from being used simultaneously. Finally, RT-Select makes radio selection decisions based on three different cases:

1. if the more energy-efficient radio R_{idx_1} can deliver all packets before the deadline by itself, RT-Select uses R_{idx_1} alone and assigns all N packets to it. (Line 4-5)

2. if none of the radios can deliver all packets before the deadline by itself, RT-Select attempts to use both radios. First, RT-Select assigns $X_{max\_(idx\_1')}$ packets to $R_{idx\_1'}$. Then, the remaining packets are assigned to the other radio if there is no hardware conflict between the two radios. (Line 6-10)

3. if only the less energy-efficient radio R_{idx_2} can deliver all packets before the deadline, RT-Select needs to decide whether to use it alone or use both radios. In case R_{idx_2} has the smaller B_i of the two radios or $X_{max\_(idx\_1')}$ is smaller than $A_{idx\_1'}/(B_{idx\_2} - B_{idx\_1'})$ [3], RT-Select uses the less energy-efficient radio R_{idx_2} alone and assigns all N packets to it. If there exists a hardware conflict between the two radios, R_{idx_2} is also used alone to avoid the conflict. Otherwise, RT-Select selects both radios and assigns

[3]This comparison decides whether it consumes less energy to use the less energy-efficient radio alone. The equation comes from the optimality proof in [3].

$X_{max\_(idx\_1')}$ packets to $R_{idx\_1'}$ and the remaining packets to the other radio. (Line 12-17)

4.3.2 RT-Select-General Algorithm for Selection among Multiple Radios

Algorithm 3: RT-Select-General

```
Input  : $N, D, RC_1, RC_2, ..., RC_m$
Output: $X_1, X_2, ..., X_m$
1  Compute $\{A_i, B_i, X_{max\_i} \mid i = 1, ..., m\}$ ;
2  $(idx\_1, ..., idx\_m) = \textbf{sort}\{A_i + B_i \times N \mid i = 1, ..., m\}$ ;
3  $(idx\_1', ..., idx\_m') = \textbf{sort}\{B_i \mid i = 1, ..., m\}$ ;
4  if $X_{max\_(idx\_1)} \geq N$ then
5      $X_{idx\_1} \leftarrow N$ ;
6  else if $\textbf{max}\{X_{max\_(idx\_i)} \mid i = 1, ..., m\} < N$ then
7      for $i = 1$ to $m$ do
8          if Conflict($R_{idx\_i'}, \{R_k \mid X_k > 0\}$) then
9              continue;
10         end
11         if $X_{max\_(idx\_i')} < N - \textbf{sum}\{X_{idx\_k} \mid k < i\}$ then
12             $X_{idx\_i'} \leftarrow X_{max\_(idx\_i')}$ ;
13         else
14             $X_{idx\_i'} \leftarrow N - \textbf{sum}\{X_{idx\_k} \mid k < i\}$ ;
15             break;
16         end
17     end
18 else
19     for $i = 2$ to $m$ do
20         if $X_{max\_(idx\_i)} < N$ then
21             continue;
22         end
23         if $B_{idx\_i} = B_{idx\_1'}$ or $A_{idx\_1'}/(B_{idx\_i} - B_{idx\_1'}) > X_{max\_(idx\_1')}$ or
             Conflict($R_{idx\_i}, R_{idx\_1'}$) then
24             $X_{idx\_i} \leftarrow N$ ;
25         else
26             $X_{idx\_1'} \leftarrow X_{max\_(idx\_1')}$ ;
27             $X_{idx\_i} \leftarrow N - X_{idx\_1'}$ ;
28         end
29         break;
30     end
31 end
```

Based on the insights collected during our algorithm design for the two-radio special case, we design RT-Select-General that solves the general form of the problem involving m radios. As shown in Algorithm 3, RT-Select-General first computes the A_i, B_i, and X_{max_i} values for all m radios (Line 1). Similar to RT-Select, RT-Select-General sorts all m radios based on the energy consumption to transmit N packets for each single radio ($A_i + B_i \times N$) and stores the sorted radio indices to $(idx\_1, ..., idx\_m)$ in ascending order (Line 2). RT-Select-General sorts all radios again based on the average energy consumption per packet B_i without considering radio switching energy consumption A_i and stores the radio indices to $(idx\_1', ..., idx\_m')$ in ascending order (Line 3). The radio hardware conflict checker "*Conflict*(R_x, R_y)" gets the boolean information on whether there is a hardware conflict between the radio R_x and any radio in R_y, where R_y is a set that consists of one or more radios.

RT-Select-General makes radio selection decisions based on three cases similar to RT-Select:

1. if the most energy-efficient radio R_{idx_1} can deliver all packets before the deadline by itself, RT-Select-General uses it alone and assigns all N packets to it. (Line 4-5)

2. if none of the radios can deliver all packets before the deadline by itself, RT-Select-General has to use multiple radios. Similar to RT-Select, RT-Select-General prefers to use the radios with small B_is, thus it selects the radios one by one based on the sorted indices $(idx\_1', ..., idx\_m')$ and lets them transmit with their maximum capacity until the selected radios can deliver all N packets before the deadline. If there exists a radio hardware conflict between $R_{idx\_i'}$ and any radio R_k which has already been selected

($X_k > 0$), the radio $R_{idx\_i'}$ is skipped to avoid the conflict. (Line 6-17)

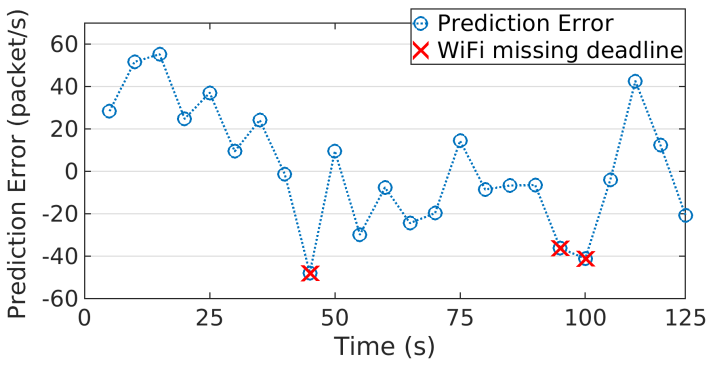
3. if there exists a radio R_{idx_i} which can deliver all packets before the deadline by itself but is not the most energy-efficient one ($i > 1$), then RT-Select-General needs to decide whether to use it alone or combine it with another radio[4]. Inspired by Algorithm 2, we consider the radio $R_{idx\_1'}$ (the one with the smallest B_i of all radios) for the possible combination with R_{idx_i}. If R_{idx_i} has the smallest B_i or $X_{max\_(idx\_1')}$ is smaller than $A_{idx\_1'}/(B_{idx\_i} - B_{idx\_1'})$, RT-Select-General selects R_{idx_i} only and assigns all packets to it. If there exists a hardware conflict between R_{idx_i} and $R_{idx\_1'}$, R_{idx_i} is also selected to be used alone. Otherwise, RT-Select-General combines R_{idx_i} with $R_{idx\_1'}$ and let $R_{idx\_1'}$ transmit with its maximum capacity and assigns the remaining packets to R_{idx_i}. (Line 19-30)

The constraints reflecting the hardware conflicts can be added into case 2) and case 3) of Algorithm 2 and Algorithm 3. RT-Select-General behaves identically to RT-Select when $m = 2$, making the latter a special case providing optimal selections. The time complexity of RT-Select-General is $O(m \log m)$ (dominated by the complexity of sorting), which is acceptable to support real-time decision-making since m is not expected to be very large in practice ($m \leq 16$ today to our knowledge).

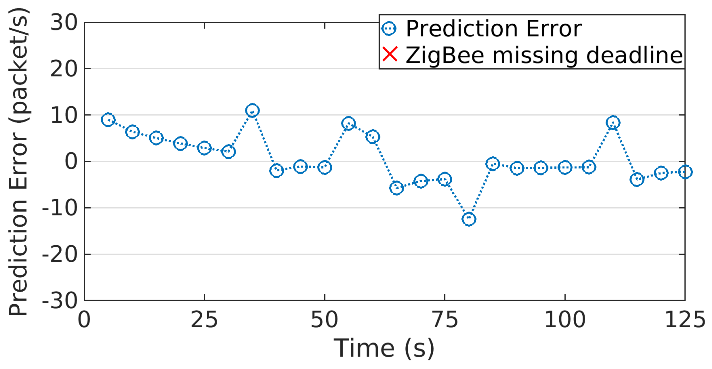
4.3.3 RT-Balance Algorithm for Runtime Traffic Balancing

As discussed in Section 4.3.1 and Section 4.3.2, RT-Select and RT-Select-General are designed to ensure that all packets can be delivered to their destina-

[4]We select at most two radios in this case in consideration of designing a light-weight algorithm for runtime use.



(a) WiFi link.



(b) ZigBee link.

Figure 4.1: Throughput prediction errors. The deadline misses are marked in red.

tion by the deadline if they can find feasible radio selection and data partitioning solutions with the assumption that the actual runtime throughput follows the predicted value TH_i. In reality, there does not exist any throughput predictor which achieves 100% prediction accuracy. To study the impact of inaccurate throughput prediction, we perform an empirical study. We use Holt-Winter predictor [81], one of the most effective time series forecasting algorithms, to predict throughput based on historical measurements, run RT-Select to select radios and partition the traffic, and record the deadline misses. We observe that a deadline miss occurs when the traffic assigned to the radio R_i is close to its maximum packet delivery capacity X_{max_i} and the actual throughput of the radio R_i is smaller than the predicted value in that period. Figure 4.1 plots the throughput prediction errors when both the WiFi and ZigBee radios are selected by RT-Select to transmit 500 packets (64KB data) with a deadline of $0.8s$. Based on line 7-8 in Algorithm 2, the traffic assigned to the WiFi radio has about 478 packets, which is very close to the WiFi radio's capacity, while only about 22 packets are assigned to the ZigBee radio. As Figure 4.1(a) shows, the packet deliveries through the WiFi link miss the deadline in three periods ($45s$, $95s$, and $100s$), when the actual throughput measurements are smaller than the predictions by at least $30 packets/s$. Figure 4.1(b) shows that the packet deliveries through the ZigBee link always meet the deadline because the traffic assigned to the ZigBee radio is far below its capacity. From the results, we can see that the deadline misses occur when the traffic assigned to a radio is very close to its capacity.

Algorithm 4: RT-Balance

```
Input      : $N, D, RC_1, ..., RC_m$
Global Var: $seq \leftarrow 0$
Compute $\{X_{max\_i} \mid i = 1, ..., m\}$;
if $\sum_{i=1}^{m} X_{max\_i} > N$ then
    goto RT-Select(-General) ;
end
for $i = 1$ to $m$ do
    if fork() $> 0$ then
        continue ;
    end
    while $seq < N$ do
        if isReady ($R_i$) then
            Tx ($R_i$, ++seq) ;
        end
        if time() $> D$ then
            return FAIL ;
        end
    end
    return OK ;
end
```

To address this issue, we reserve a small portion of the predicted throughput (e.g., 5%) as a guard space, compute X_{max_i} based on the rest (e.g., 95%), and design a runtime algorithm, namely RT-Balance, which balances the traffic assigned to different radios. Algorithm 4 shows the RT-Balance algorithm. When facing tight deadlines, RT-Balance creates a process for each radio that repeatedly transmits a packet when it is ready (Line 5-18). In this way, RT-Balance minimizes the latency to meet the deadline and achieves natural load balance among the radios. Specifically, a global variable "*seq*", storing the sequence number of the current packet assigned for transmission, is shared by all processes and initialized as 0. Algorithm 4 first computes the packet delivery capacity (X_{max_i}) of each radio R_i (Line 1), where only the radios without hardware conflict are considered. Then, if the sum of all radios' packet delivery capacities is larger than the traffic demand, RT-Select or RT-Select-General is used to select radios and partition data (Line 2-3). Otherwise, the load balancing is invoked and m child processes are created for the m radios using "*fork*()" (Line 5-7). Each child process uses a loop to request packets for transmission until all packets have been assigned. If there is any unassigned packet and the radio R_i is ready to transmit, *seq* is incremented to be the sequence number of a new packet, which is assigned to the radio R_i for transmission (Line 9-11). The time that has passed since the program starts is checked in each loop. If the deadline has passed before all packets have been transmitted, the child process terminates and indicates that the deadline has been missed (Line 13-14). Otherwise, the child process finishes after all transmission is complete (Line 17).

4.4 System Design and Implementation

To realize our designs, we develop the RRaSB system that makes multiple radios available at runtime and allows dynamic radio switching and bundling among them. Figure 4.2 shows the system architecture. The radio characteristics including energy consumption of radio switching (E_{sw}), radio switching time (T_{sw}), power consumption when the radio is idle (P_{rb}), and average energy consumption per transmission attempt (E_{ta}) are measured offline and stored in the **Radio Characteristics** component, serving as inputs to the radio selection algorithm. The **Throughput Predictor** predicts the throughput in the next period based on the historical data and the **Link Quality Predictor** estimates the expected transmission counts (ETX) in the next period based on previous ETX measurements using the Holt-Winters method. If a radio has not been used for a long time, Link Quality Predictor transmits some probing packets through it to keep its link quality measurements updated. The **Radio Selection Engine** takes radio characteristics, estimated throughput and ETX, and traffic demand and deadline specified by the application as inputs and runs the radio selection algorithm to select the radio(s) that is/are best suited for the current network traffic and operating conditions and then assigns packets accordingly. Multiple **Radio Controller** modules exist in RRaSB. Each Radio Controller controls the on/off state of a radio based on the decision made by the Radio Selection Engine and measures the actual throughput and ETX fed into the Throughput Predictor and Link Quality Predictor, respectively. RRaSB is configured to perform the radio selection in each period based on the measured throughput and ETX of the radio links as well as the traffic demand and deadline specified by the benchmark appli-

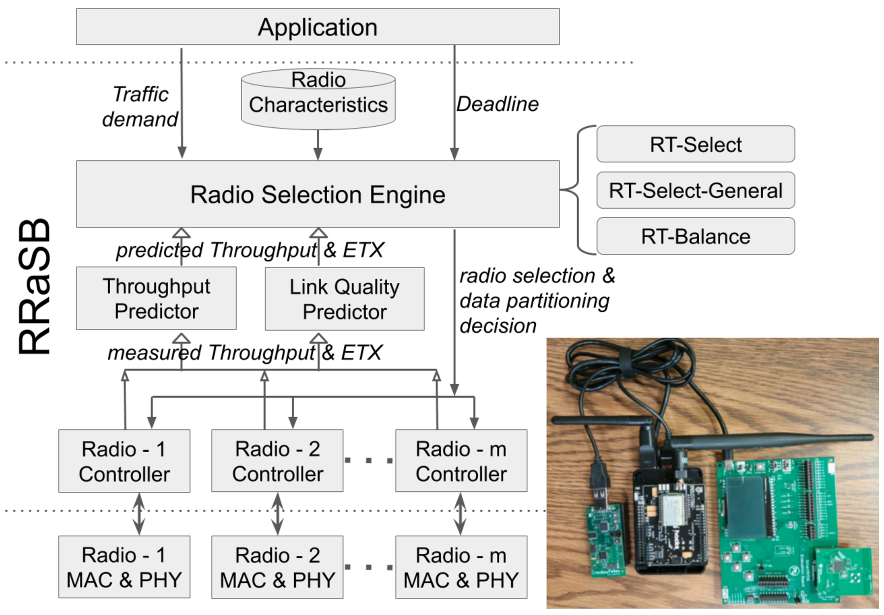


Figure 4.2: System architecture and the platform supporting five radios.

cation. If the current radio selection is found to be the best-suited, it is retained; otherwise, our system switches to a new best-suited setting. Radios are turned off after the last transmission in each period if they are not selected for use in the next period and the unselected ones are kept off to reduce energy consumption. If multiple transmitters exist, they access the channel in a TDMA fashion. We have implemented RRaSB in Raspbian Linux [63] and Contiki [64] and two prototypes: one with two radios and the other with five radios. A power monitor from Monsoon Solutions [62] is connected to the sender to measure the energy consumption. More implementation details can be found in [3].

4.5 Evaluation

To examine the efficacy of our radio selection and traffic partitioning solution, we perform a series of experiments on our embedded platform presented in Section 4.4. We start by demonstrating the time efficiency of RT-Select-General and the effectiveness of the throughput and link quality predictors. We then run experiments to measure the radio energy consumption and deadline miss ratio with our prototype hosting two radios and repeat the experiments with five radios. We compare our approaches against two baselines: GreenBag using GB-E configuration [11] and GLPK (GNU Linear Programming Kit) [82]. GreenBag is a practical state-of-the-art radio selection approach designed for real-time applications. GreenBag supports multi-radio mode and single-radio mode under GB-E and GB-P configurations. In multi-radio mode, GreenBag seeks to minimize the transmission time by balancing the load on multiple radios based on link throughput prediction, while the most energy-efficient radio is selected in single-radio mode. GB-E chooses single-radio mode to reduce the energy consumption and switches to multi-radio mode when the bandwidth is insufficient, while GB-P uses multi-radio mode only. GLPK provides the optimal results to the ILP problems. Please note that GLPK cannot be used for real-time applications with short deadlines because of its heavy computation overhead as presented in Section 4.5.1. We run GLPK offline and exclude its energy consumption in the results of optimal solutions (Figure 4.8(a) and 4.9(a)).

In all experiments, we deploy two real-time benchmark applications on top of our system which generate data packets periodically. The first benchmark application (benchmark application A) emulates a health care scenario where doctors use smart glasses to take ambient pictures or videos of patients and send them to the hospital information system for real-time face detection and recognition [74]. In this application, a fixed traffic demand is employed by the smart glasses but the application may specify different deadlines based on its quality of service (QoS) needs. The second benchmark application (benchmark application B) emulates a

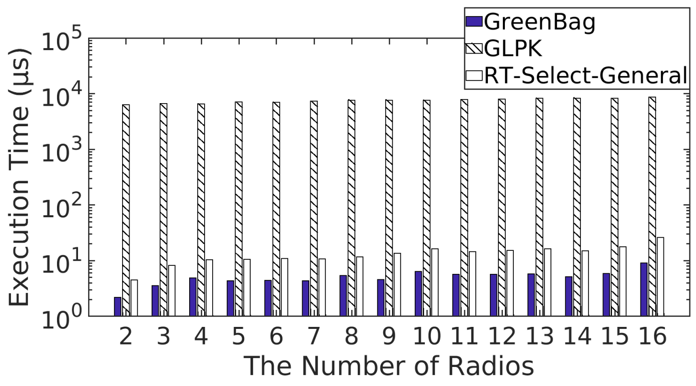


Figure 4.3: Execution time of RT-Select-General compared with GreenBag and GLPK.

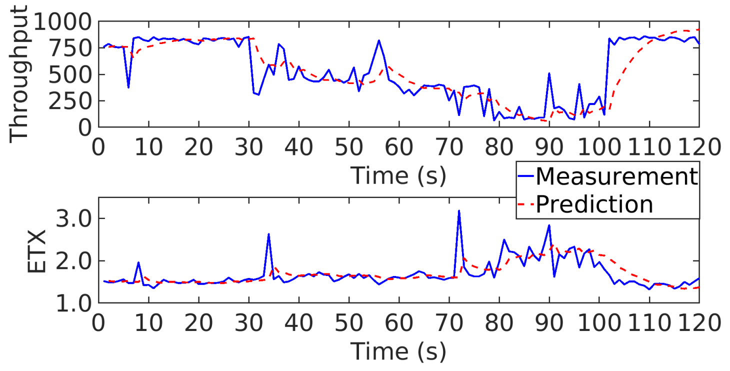


Figure 4.4: Throughput and ETX predictions vs. ground truth in a 120-second WiFi link condition trace.

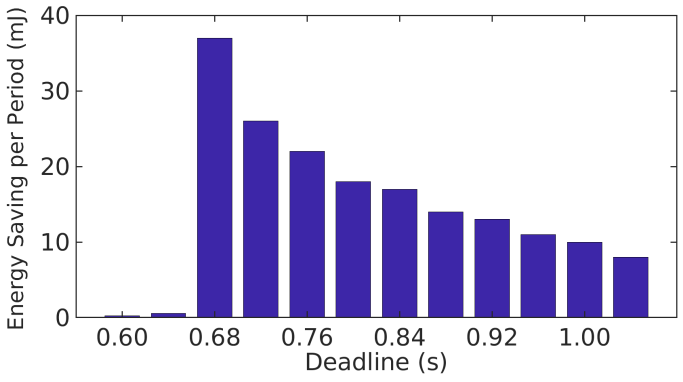
real-time georeferencing scenario where UAVs capture images of the land from the air and transmit them together with GPS locations to a ground station [75]. In this application, a fixed deadline (e.g., 1 second) of image delivery is adopted by the UAVs to ensure the accuracy of the real-time location but the traffic demand (image size) may vary to meet different needs. Both benchmark applications generate periodic traffic whose deadline is equal to its period. The two benchmark applications allow us to examine the performance of our system (i) at a fixed data rate with different data delivery deadlines and (ii) at various data rates with a fixed deadline.

4.5.1 Time Efficiency of RT-Select-General

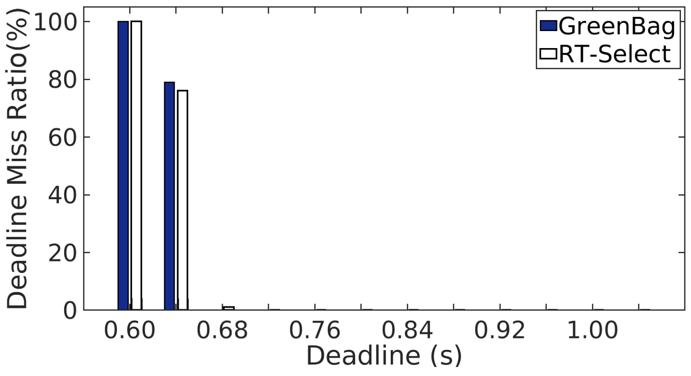
We first measure the execution time of RT-Select-General and two baseline approaches (GreenBag and GLPK) on the Raspberry Pi 3 with a 1.2 GHz 64-bit quad-core ARMv8 CPU. We measure the time duration between feeding the input into the Radio Selection Engine and receiving the output from it. We repeat the experiments 20 times using random inputs for each m (the number of radios). Figure 4.3 shows the average execution time of GreenBag, GLPK and RT-Select-General for different number of radios (m ranging from 2 to 16) in the logarithmic scale. As Figure 4.3 shows, the average execution time of RT-Select-General increases from $4\mu s$ to $26\mu s$ when m increases from 2 to 16, which is slightly ($2{\sim}17\mu s$) longer than what GreenBag uses. The average execution time of GLPK ranges from $6267\mu s$ to $8670\mu s$, which is 336∼1412 times longer than what RT-Select-General consumes. Therefore, it is not feasible to use the time-consuming GLPK to support the real-time applications with short deadlines, especially when running on the platforms with limited harware resources. As a comparison, our RT-Select-General can time-efficiently make decisions achieving performance close to what GLPK offers (see Section 4.5.4).

4.5.2 Effectiveness of Link Condition Predictors

We then perform a set of controlled experiment to evaluate the effectiveness of our Throughput Predictor and Link Quality Predictor employing the Holt-Winters method. In this set of experiments, we measure the throughput and ETX of radio links under controlled interference and compare them against the predicted values. Figure 4.4 plots the example traces showing the throughput and ETX changes of a



(a) Energy saving over GreenBag.



(b) Comparison on deadline miss ratio.

Figure 4.5: Performance under RT-Select and GreenBag with two radios when the application transmits at a fixed data rate with different deadlines.

WiFi link when encountering the controlled interference. An interferer begins the transmission in the same channel from the 31st second to the 100th second. As Figure 4.4 shows, the predictions are very close to the measurements during the process. The standard deviation on the throughput difference is 152 packets/s and 80% of the prediction errors are less than 125 packets/s. The standard deviation on the ETX difference is 0.25 and 80% of the prediction errors are less than 0.2.

4.5.3 Experiments with Two Radios

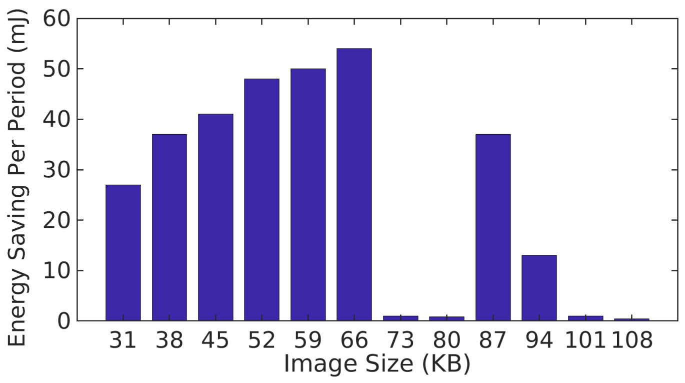
We run experiments on our prototype hosting two radios [3] (i.e., the CC2650 ZigBee radio and the RT5370 WiFi radio) to evaluate the effectiveness of RT-Select and its impact on radio energy consumption and real-time performance.

Since the output of RT-Select is proved to be optimal, we only compare RT-Select against GreenBag in this set of experiments.

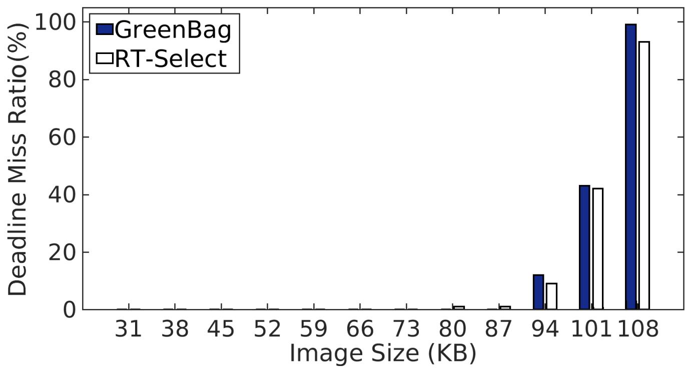
We configure the benchmark application A to transmit a 23KB image (480×480 JPEG) in every period and repeat the experiments with 12 different deadlines ranging from $0.60s$ to $1.04s$ according to the response time of Amazon face recognition applications [83]. Figure 4.5(a) shows the energy saving of RT-Select over GreenBag per period and Figure 4.5(b) plots the deadline miss ratio. RT-Select shows significant energy saving (ranging from $8mJ$ to $37mJ$[5]) when the deadline is greater than $0.64s$ with the deadline miss ratios no higher than 1%. The energy savings benefit from RT-Select's decision on keeping only the WiFi radio active rather than using both radios suggested by GreenBag. High deadline miss ratios are observed under both RT-Select and GreenBag when the deadline is shorter than $0.68s$, not enough to turn on the WiFi radio or send all packets using the ZigBee radio. The results show that RT-Select consistently outperforms GreenBag under various deadlines.

Similarly, we configure the benchmark application B to transmit a JPEG image with the fixed deadline ($0.80s$) in every period, and repeat the experiments with 12 image sizes ranging from 31KB (640×480 JPEG) to 108KB (1280×720 JPEG). As Figure 4.6(a) and Figure 4.6(b) show, RT-Select consumes $27{\sim}54mJ$ less energy compared to GreenBag without missing any deadline when the image size is between 31KB and 66KB. The energy savings benefit from RT-Select's decision on keeping only the WiFi radio active rather than using both radios suggested by GreenBag. The energy saving is marginal when the image size is 73KB or 80KB.

[5]As a comparison for energy saving values, the CC2650 radio consumes $30mW$ power when transmitting at $5dBm$ [58].



(a) Energy saving over GreenBag.

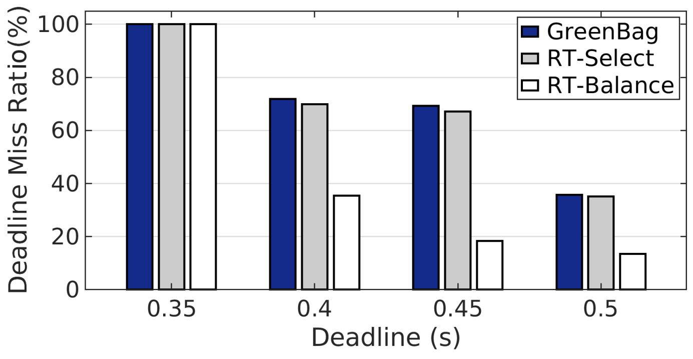


(b) Comparison on deadline miss ratio.

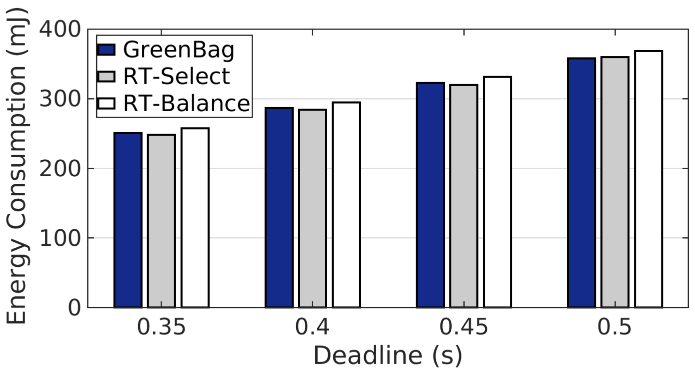
Figure 4.6: Performance under RT-Select and GreenBag with two radios when the application transmits at different data rates with the same deadline.

This is because both RT-Select and GreenBag decide to use only the WiFi radio when it becomes the more energy-efficient radio under high traffic demand and can deliver all data packets by the deadline. When the image size is 87KB, both RT-Select and GreenBag suggest using both radios. However, RT-Select assigns 94.6% of packets to the WiFi radio and 5.4% to the ZigBee radio and lets WiFi transmit for the entire period and ZigBee finish early, while GreenBag assigns 85.9% of packets to the WiFi radio and 14.1% to the ZigBee radio and lets both radios finish their transmissions at the same time, resulting RT-Select consumes $37mJ$ less energy than GreenBag. High deadline miss ratios are observed under both RT-Select and GreenBag when the image size is larger than 87KB, beyond the capacity of two radios with the consideration of radio switching overhead. The results show that RT-Select always provides the better radio selections on various data rates.

To evaluate the performance of RT-Balance, we configure the benchmark application A to transmit a fix sized image of 64KB with some tight deadlines ranging from $0.35s$ to $0.50s$. Since the deadlines are very tight, both radios have to keep active for the entire period. As Figure 4.7(a) and 4.7(b) show, RT-Balance significantly reduces the deadline miss ratio by 34.5%, 48.9% and 21.7% compared to RT-Select when the deadlines are $0.40s$, $0.45s$ and $0.50s$, respectively. At these deadlines, RT-Balance only increases the energy consumption by $11mJ$, $12mJ$ and $8mJ$ per period. The slight increase in energy consumption is in exchange for a proportionally much-larger reduction in deadline miss ratio. The reduction on the deadline miss ratio benefits from RT-Balance's runtime traffic balancing between the two radios, in contrast to RT-Select and GreenBag which assign packets to



(a) Comparison on deadline miss ratio.



(b) Comparison on energy consumption.

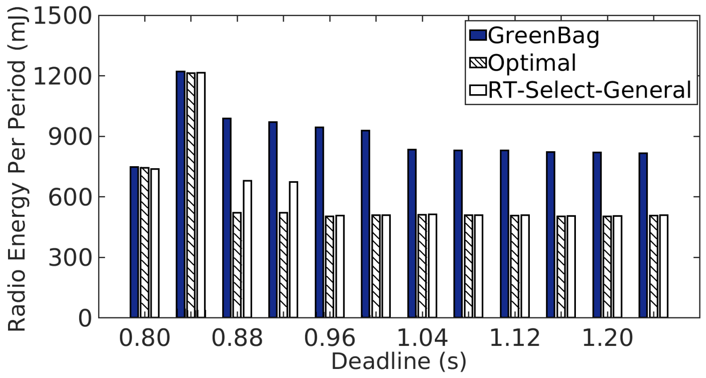
Figure 4.7: Performance of GreenBag, RT-Select, and RT-Balance with two radios when the application transmits at a fixed data rate with different deadlines.

each radio before transmission based on throughput prediction. The deadline miss ratios are 100% for all approaches when the deadline is 0.35s, which is too short for the two radios.

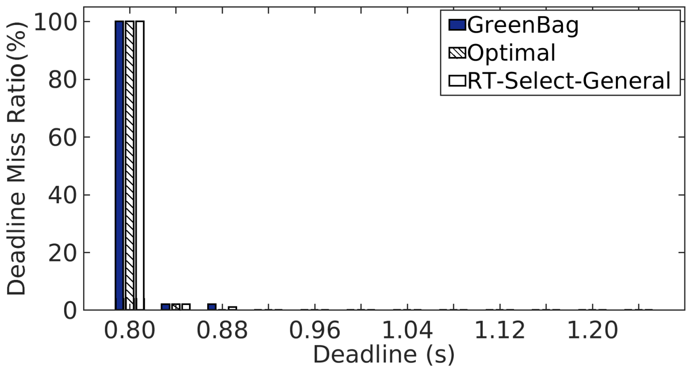
4.5.4 Experiments with Five Radios

In this set of experiments, we examine the effectiveness of RT-Select-General with our prototype device hosting five radios [3]. We compare RT-Select-General against GreenBag and Optimal.

We first explore RT-Select-General's performance under a fixed traffic demand with different deadline requirements. We configure the benchmark application A to transmit a 109KB image (1280×720 JPEG) in each period and repeat the experiments with 12 different deadlines ranging from 0.80s to 1.24s. Figure 4.8



(a) Comparison on energy consumption.

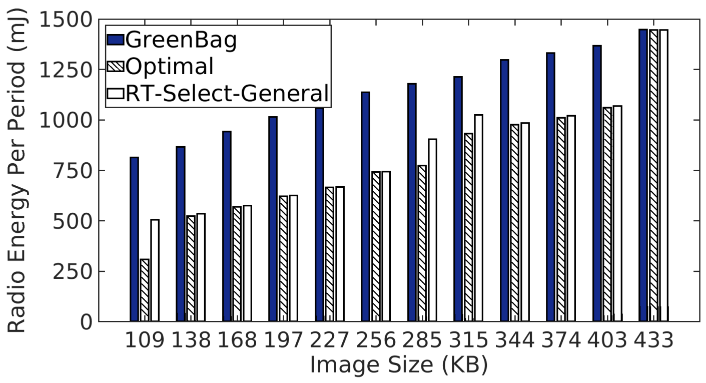


(b) Comparison on deadline miss ratio.

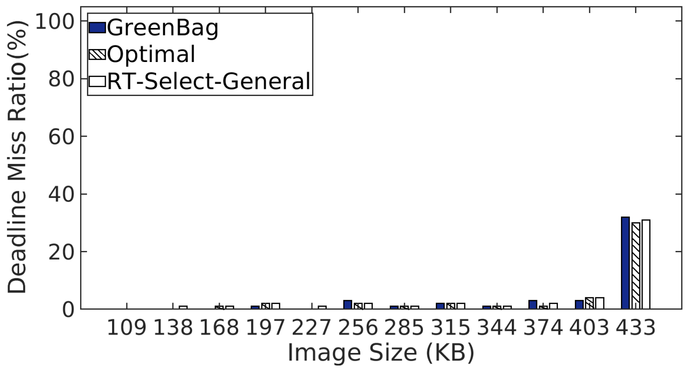
Figure 4.8: Performance of GreenBag, Optimal and RT-Select-General solutions with five radios when the application transmits at a fixed data rate with different deadlines.

shows the comparisons on radio energy consumption and deadline miss ratio under GreenBag, Optimal, and RT-Select-General, respectively. As Figure 4.8(a) and Figure 4.8(b) show, all three methods suggest using all radios to accommodate the tight deadlines (i.e., $0.80s$ and $0.84s$). High deadline miss ratios are observed when the deadline is $0.80s$, beyond the capacity of all five radios together when considering radio switching overhead. When the deadline is larger than $0.84s$, RT-Select-General achieves significant energy savings ranging from $308mJ$ to $436mJ$ compared to GreenBag with the deadline miss ratios no higher than 1%. RT-Select-General makes the optimal selections for all deadlines except $0.88s$ and $0.92s$. In those two cases, RT-Select-General selects to use the BCM43438 radio as the secondary radio based on the sorting of B_i (see Section 4.3.2), while Optimal decides to use the CC2420 radio instead.

We also evaluate RT-Select-General's performance under various traffic demands with a fixed deadline. We configure the benchmark application B to transmit a JPEG image with a fixed deadline ($1.44s$) in each period and repeat the experiments with 12 different image sizes ranging from 109KB (1280×720 JPEG) to 433KB (1920×1080 JPEG). As Figure 4.9(a) shows, RT-Select-General consistently consumes less energy ($298mJ$ on average) compared to GreenBag and performs close to what Optimal offers ($30mJ$ difference on average). RT-Select-General provides optimal selections to nine cases among the 12 cases. Please note that high deadline miss ratios are observed under all three methods when the image size is 433KB, beyond the capacities of all radios operating simultaneously when considering radio switching overhead. We also perform trace-driven simulations and observe similar improvements at various combinations of traffic



(a) Comparison on energy consumption.



(b) Comparison on deadline miss ratio.

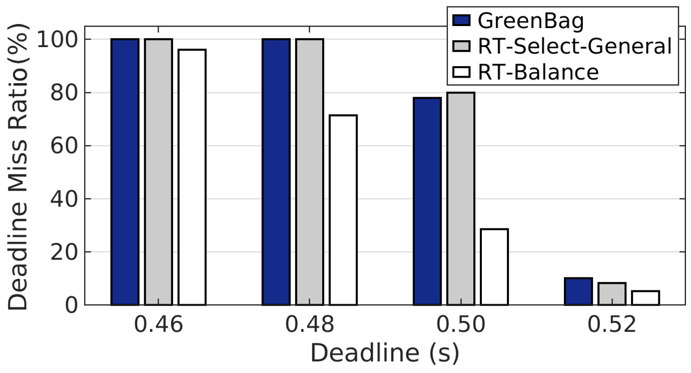
Figure 4.9: Performance of GreenBag, Optimal, and RT-Select-General with five radios when the application transmits at different data rates with the same deadline.

demand and deadline [3]. The results demonstrate the effectiveness of RT-Select-General in reducing the energy consumption, while meeting satisfactory real-time requirements.

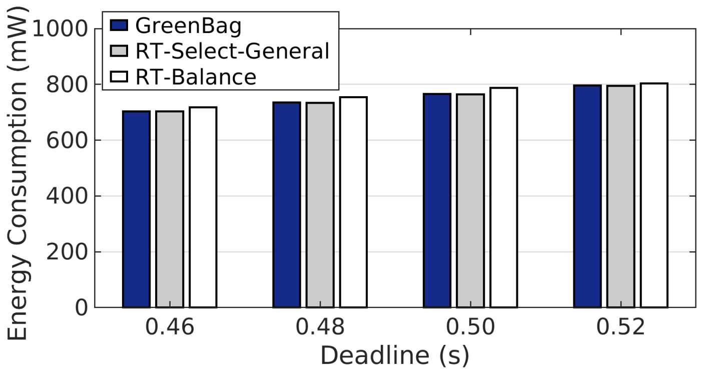
To evaluate the performance of RT-Balance, we configure the benchmark application A to transmit a fix sized image of 128KB with tight deadlines ranging from $0.46s$ to $0.52s$. Since the deadlines are very tight, all five radios have to keep active for the entire period. As Figure 4.10(a) and Figure 4.10(b) shows, RT-Balance significantly reduces the deadline miss ratio by 28.6% and 51.4% when the deadlines are $0.48s$ and $0.50s$, respectively, while only increases the energy consumption by $19mJ$ and $22mJ$ per period compared to RT-Select-General. The slight increase in energy consumption is in exchange for a proportionally much-larger reduction in deadline miss ratio. The reduction on the deadline miss ratio benefits from RT-Balance's runtime traffic balancing between the five radios, in contrast to RT-Select-General and GreenBag which assign packets to each radio before transmission based on throughput prediction. The deadline miss ratios are nearly 100% for all approaches when the deadline is $0.46s$, which is too short for the five radios.

4.5.5 Large-Scale Simulation Study

Relying on the radio characteristics measured on our platform with five radios, we also perform a large-scale simulation study to measure radio energy consumption and deadline miss ratio at various combinations of traffic demands and deadlines. In this set of experiments, we uniformly select 200 image sizes ranging from 94KB (1280×720 JPEG) to 847KB (3840×2160 JPEG) and 200 deadline samples ranging from $0.8s$ to $2.6s$ and then simulate radio energy consumption of running

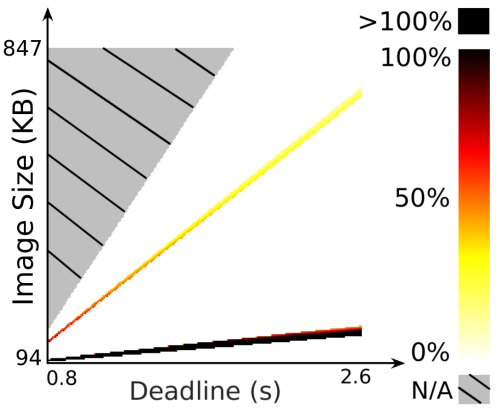


(a) Comparison on deadline miss ratio.

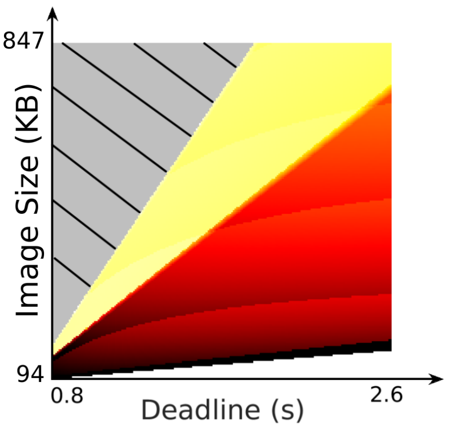


(b) Comparison on energy consumption.

Figure 4.10: Performance of GreenBag, RT-Select-General, and RT-Balance with five radios when the application transmits at a fixed data rate with different deadlines.



(a) RT-Select-General over Optimal.



(b) GreenBag over Optimal.

Figure 4.11: Radio energy comparisons with five radios at various combinations of traffic demands and deadlines. The grey shaded areas denote the invalid combinations that the optimal deadline miss ratio is higher than 5%. The colors in each subfigure denote the percentages of more energy consumed than Optimal, i.e., $(E(RT\_Select\_General) - E(Optimal))/E(Optimal)$ and $(E(GreenBag) - E(Optimal))/E(Optimal)$, respectively.

Optimal, GreenBag, and RT-Select-General, respectively, under all valid combinations of traffic demands and deadlines (optimal deadline miss ratio no higher than 5%).

Figure 4.11(a) is a heat map plotting the energy consumption difference between RT-Select-General and Optimal and Figure 4.11(b) shows the difference between GreenBag and Optimal. The white areas of Figure 4.11(a) shows the cases (94.4% of deadline and image size combinations) where RT-Select-General makes the optimal radio selections and traffic partitions. GreenBag only makes the optimal decisions in 5.4% of combinations, as shown in Figure 4.11(b). The mean energy consumption difference between RT-Select-General and Optimal is 7.1%, while the difference between GreenBag and Optimal is 60.8%. The simulation results confirm that RT-Select-General can provide optimal selections to most cases and significantly outperforms GreenBag under various combinations of data rates and deadlines.

4.6 Conclusion

Heterogeneous radios are becoming increasingly available in modern embedded devices, offering new opportunities to use multiple wireless technologies energy-efficiently to accommodate the needs of real-time applications. We formulate the runtime radio switching and bundling for real-time IoT applications as an optimization problem and present three algorithms which select radios and partition data at runtime to minimize the energy consumption for real-time data transfer. Experimental results show that the proposed solution can significantly reduce the radio energy consumption over the state of the art, while meeting the application specified traffic demand and deadline requirement.

5 Runtime Control of LoRa Spreading Factor for Campus Shuttle Monitoring

5.1 Introduction

Satellite and cellular technologies are traditionally used to collect real-time data from running vehicles to the base station through their long-distance links. For instance, LTE-based communication systems have been integrated into the urban transit systems [26, 27], while satellite links have been used to support communication among emergency vehicles [25]. However, such systems are often costly because they use expensive devices and licensed frequency bands, which prevents them from being used in many applications. As an emerging Low-Power Wide-Area Network (LPWAN) technology, LoRa is a low-cost alternative that can support long-range data collection for low data rate applications [29, 35]. For a small service area, such as a university campus, LoRa offers a cost-effective communication solution because one or a few base stations are enough to cover the area and battery-powered LoRa modules can inexpensively retrofit existing devices.

In this chapter, we introduce the *ShuttleNet*, a LoRa-based wireless networking solution that collects real-time data from shuttles that circle a university campus using a fixed route. Multiple types of data, such as vehicle speed, the vehicle's operating condition, and the number of passengers, are collected to enhance the safety and efficiency of shuttle service and improve rider experience. For instance, the vehicle speed information is used to estimate the expected time of arrival (ETA) at each shuttle stop. The number of passengers is used to monitor the transit demand, which allows more shuttles to be dispatched when needed. Warnings or alarms are generated if the vehicle's operating condition degrades. The data needed to be collected falls into one of two categories: time-critical and non-time-critical. The time-critical data, such as the current vehicle speed and the number of passengers, needs to be collected in real time with high reliability because the data may become useless if it fails to be delivered in time. The non-time-critical data, such as the data reflecting the vehicle's operating condition (e.g., accelerating and braking performance), can be delayed because the operating condition of a vehicle does not change very frequently. Similar to the tradeoff between network reliability and throughput in cellular networks [84], the tradeoff also exists in LoRa networks, which results in a significant challenge on the selection of the LoRa Spreading Factor (SF). A larger SF provides higher reliability at the cost of lower throughput and vice versa. The fluctuations of low-power LoRa link quality resulting from the mobility of vehicles significantly amplify the challenge. Therefore, there exists a critical need for a new solution that makes good tradeoffs between network reliability and throughput for LoRa-based mobile wireless networks.

In this chapter, we present a low-cost LoRa-based networking solution that employs a novel runtime SF control solution to maximize the data collection throughput from running vehicles while meeting the reliability requirement specified by the application. Specifically, we make the following contributions:

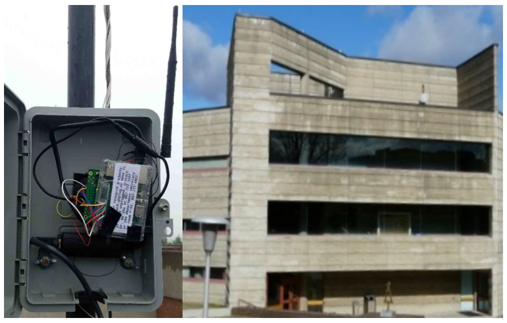
- We present a low-cost LoRa-based wireless networking solution that collects real-time data from six shuttles circling our university campus;
- To our knowledge, this is the first solution to investigate the SF selection for LoRa devices installed on running vehicles, distinguished from previous work designed for stationary devices. The study has been performed in the real world for more than a year;
- We provide a practical runtime LoRa SF control solution that employs the K-Nearest Neighbors (KNN) algorithm and meets network performance requirements with small computation overhead;
- Our experimental results show that our runtime LoRa SF control solution significantly outperforms the state-of-the-art methods.

The remainder of the chapter is organized as below. Section 5.2 introduces the background of LoRa. Section 5.3 presents our design of ShuttleNet. Section 5.4 describes our empirical study of LoRa SF configurations. Section 5.5 presents the design of our runtime SF control solution, and Section 5.6 compares its performance against three baselines. Section 5.7 concludes the chapter.

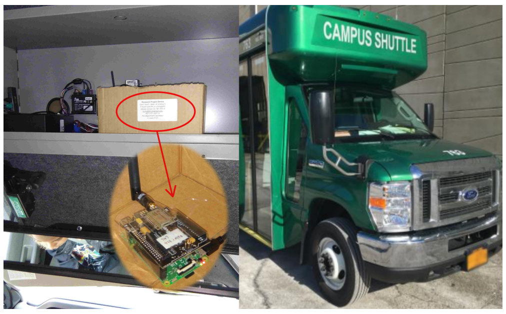
5.2 Background

Recently, LoRa has emerged as a popular LPWAN technology that provides long-range communication. LoRa employs the Chirp Spread Spectrum (CSS) modulation, where a LoRa signal, namely a chirp, increases or decreases its operating frequency linearly through time and circularly sweeps through its predefined frequency band. The time duration of transmitting a chirp depends on the selections of the LoRa physical-layer parameters: SF and Bandwidth (BW). The reliability of a LoRa link is measured by the Packet Delivery Ratio (PDR), which depends on the Signal-to-Noise Ratio (SNR) and the Received Signal Strength (RSS) at the receiver. A larger SF allows the receiver to receive a packet with a lower RSS by exponentially expanding the duration of each chirp, which results in a lower data rate. A chirp that uses the SF value of x can represent x bits of data and takes the time duration of $2^x/BW$ to be transmitted [85]. With a fixed BW, the data rate of a chirp is proportional to $(x \cdot 2^{-x})$, which decreases nearly exponentially with x. There are six SF configurations available in the sub-GHz ISM bands (from $SF7$ to $SF12$). The data rates under $SF8$ to $SF12$ are 57.1%, 32.1%, 17.9%, 9.8%, and 5.4% of the one under $SF7$, respectively. The selection of SF decides the data rate and communication range of a LoRa link. The LoRa signal that carries a packet consists of a series of chirps, where the preamble chirps are followed by the data chirps. The preamble chirps are used by the receiver to identify the start of a LoRa packet. The data chirps contain up to 255 bytes including the packet header and payload sections. Both sections end with a Cyclic Redundancy Check (CRC) code to verify the integrity of the received data. The payload section contains the data bits with some inserted Hamming Code redundant bits for error correction. The Coding Rate (CR) of Hamming Code (e.g., “4/5”) is defined as a fraction in which the numerator is the number of data bits and the denominator is the total number of data bits and redundant bit(s). The CR setting of a packet is stored in its header.

LoRaWAN is a Media Access Control (MAC) layer protocol designed for LoRa and defines three classes: Class A (by default), Class B, and Class C. The Class



(a) LoRa base station installed on a roof.



(b) LoRa end device installed on a shuttle.

Figure 5.1: Hardware deployment of ShuttleNet at the State University of New York (SUNY) at Binghamton.

A communication frame is initiated by an end device and consists of an uplink window (time slot) followed by two downlink windows. To send data, the end device initiates an uplink window, selects a random channel, and transmits in pure ALOHA mode [86]. The first downlink window operates on the same channel as the last uplink window and the second downlink window operates on a predefined channel [87]. Classes B and C allow downlink data to be received in a timely fashion by increasing the time that end devices listen to the channel. Class B uses periodic beacons broadcasted by the base station to open extra downlink windows at end devices while Class C keeps end devices listening at all times. In many cases, the communication frames defined in LoRaWAN cannot provide optimal performance, such as when an application only requires end devices to upload data, leaving the downlink windows unused. Moreover, theoretically speaking, the maximum throughput that pure ALOHA can provide is only 18.4% of the channel capacity [88], which limits the throughput performance of LoRaWAN. To overcome the limitations, we design new time frame and channel assignment methods for ShuttleNet (see Section 5.3.3). LoRaWAN specifies the Adaptive Data Rate (ADR) algorithm that selects SF based on the link quality measured on an end device [37]. Specifically, ADR first estimates the link quality using the maximum SNR values measured in the last 20 samples and then selects the SF based on the required SNR level for each SF [38]. Similar to the Class A frame, the ADR process is initiated by a request from an end device followed by a response from the base station with a new SF. If the response is not received by the end device before it times out, the end device increases the SF because the link is disconnected under the current SF. When an ADR request is received, the base station reduces the current SF to the most appropriate level if the estimated SNR is greater than the required SNR plus a safety margin. ADR is designed for stationary devices and does not work well on mobile devices (see Section 5.4), which motivates us to design a new runtime SF control solution for ShuttleNet.

5.3 ShuttleNet

In this section, we first present our hardware deployment and software architecture of ShuttleNet and then introduce our time frame and channel assignment designs.

5.3.1 Hardware Deployment

ShuttleNet is designed to collect data from six shuttles that circle our university campus (a 1280m × 990m area) with a fixed route. Figure 5.2 shows the route. The distance between a shuttle and our LoRa base station can be up to 860 *meters*. Figure 5.1 shows our hardware deployment. The LoRa base station is placed in a weatherproof box on the roof of a three-floor building (Figure 5.1(a)) and a LoRa end device is installed in the glove compartment above the driver seat on each of six shuttles (Figure 5.1(b)). The LoRa base station and end devices

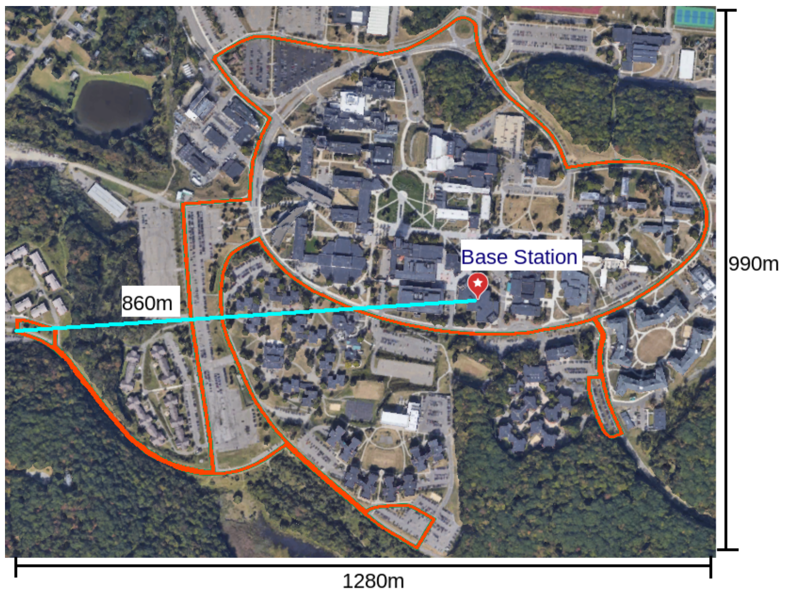


Figure 5.2: Campus shuttle route.

are built by integrating commercial off-the-shelf (COTS) devices. The LoRa base station is an embedded computer (i.e., Raspberry Pi 3 Model B) integrated with an iC980A module provided by IMST, while the LoRa end device is a Raspberry Pi computer integrated with an RN2903 module operating in the 900/915 MHz band [89]. The iC980A module is the enhanced version of iC880A [90] operating in the 900/915 MHz band. We use iC980A to build our LoRa base station because it is capable of receiving packets from multiple end devices that use different SF configurations and up to eight channels in parallel. To maximize throughput, we configure different LoRa end devices to use distinct channels. The RN2903 module can only operate on a single channel in either transmitter or receiver mode at each time. Our goal is to provide a low-cost networking solution; thus we use the RN2903 module to build the LoRa end device. Table 5.1 summarizes the costs of our devices. Please note that LoRa operates in the free, unlicensed band. The total hardware cost of ShuttleNet is $536.

Table 5.1: Price List of Hardware Components

Device Name	Price (USD)	End Device	Base Station
Raspberry Pi 3	35	√	√
RN2903 Module	13	√	
RPi Connection Bridge	8	√	
iC980A Module	130		√
Power Adapter	5	√	√
Total (USD)	536	61 × 6	170

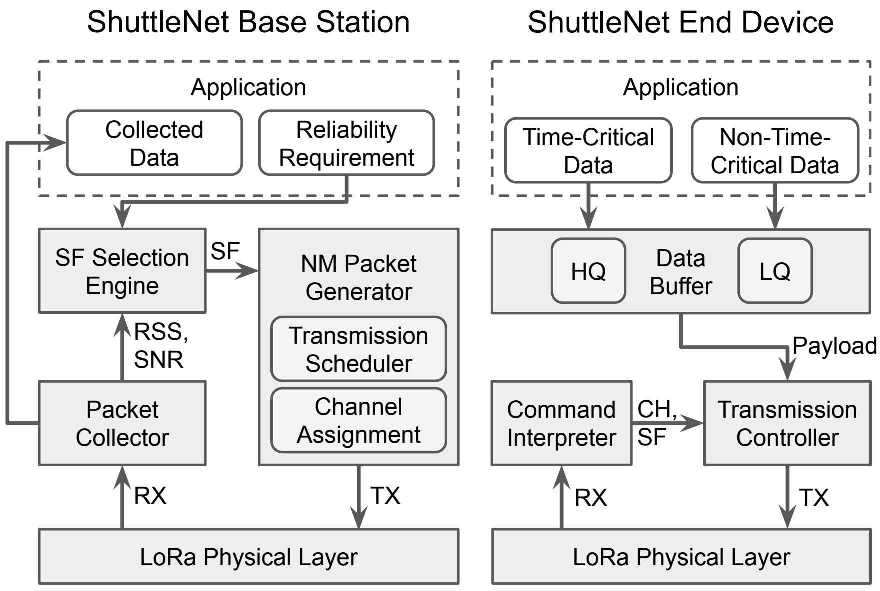


Figure 5.3: Software architecture of ShuttleNet.

5.3.2 Software Architecture

Figure 5.3 plots the software architecture of ShuttleNet. On the LoRa base station, the **Packet Collector** forwards the received packets from the LoRa physical layer to the application and collects the link characteristics including RSS and SNR for the **SF Selection Engine**. The SF Selection Engine employs our runtime SF control solution that selects the best-suited SF configuration for each end device based on the given link characteristics and the reliability requirement specified by the application (see Section 5.5). The **NM Packet Generator** broadcasts the Network Management (NM) packets, which carry the selected SF configuration, the assigned channel, and the transmission schedule for each end device (see Section 5.3.3). On each LoRa end device, the **Command Interpreter** extracts and interprets the NM commands from the received NM packets. The **Transmission Controller** transmits the data generated from the vehicle on the assigned channel using the selected SF configuration. The **Data Buffer** maintains two data queues for the data collected from the vehicle: a high-priority queue (HQ) for time-critical data and a low-priority queue (LQ) for non-time-critical data. The transmission controller only transmits the data stored in the LQ when the HQ is empty.

5.3.3 Time Frame and Channel Assignment

In ShuttleNet, the LoRa base station and end devices are time synchronized and share the notion of a time frame that repeats over time. At the beginning of each frame, the LoRa base station broadcasts a NM packet that synchronizes the clocks of all end devices. Each NM packet carries a unique frame ID and the network management information for this frame (e.g., the SF and channel assigned to each end device). ShuttleNet uses a fixed channel CH_{ctl} for downlinks (from the base station to shuttles) and assigns channels for uplinks (from shuttles to the base station) based on the number of shuttles in the network N_s. Assuming the LoRa base station can receive packets from N channels (CH_1 to CH_N) in parallel, the LoRa base station assigns a unique channel CH_i ($1 \leq i \leq N$) to each end device if $N_s \leq N$. Otherwise, the LoRa end devices share the N channels in a TDMA fashion. The NM packets carry the commands indicating which devices should transmit in each time frame. The LoRa base station always uses the physical layer parameters ($SF = 12$, $CR = 4/8$, and CRC enabled) to transmit the NM packets because the deliveries of NM packets are critical for maintaining the time

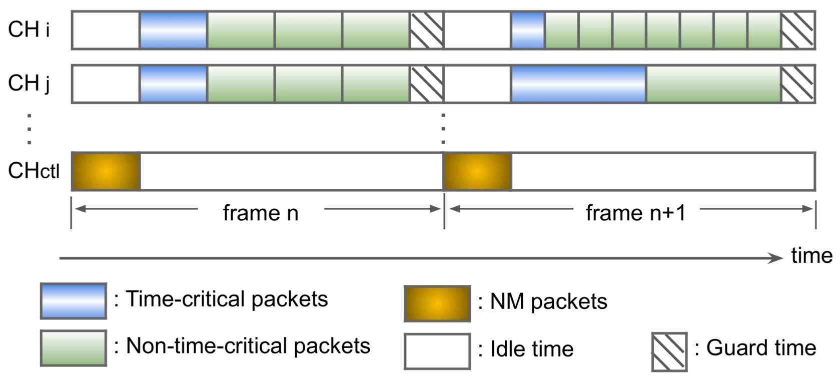


Figure 5.4: An example timeline of transmissions on different channels within two consecutive time frames in ShuttleNet.

frame.

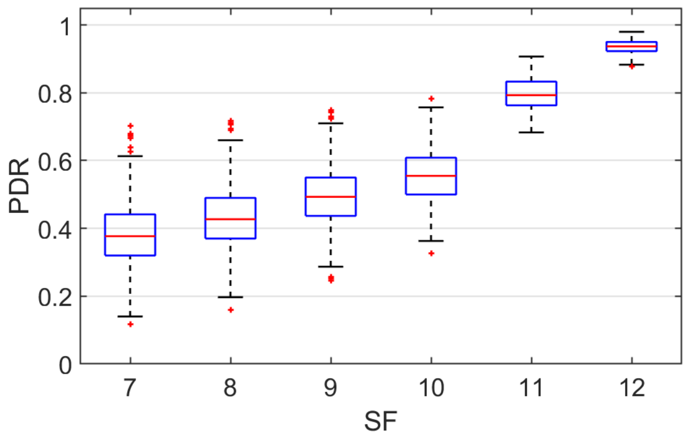
Before a LoRa end device starts to transmit, it first listens to the downlink channel CH_{ctl} and waits for the NM packet. After receiving the NM packet, the LoRa end device searches for its ID to decide whether it can transmit in this frame. If the LoRa end device is allowed to transmit, it calculates the number of packets that can be transmitted in the current frame using the assigned SF configuration. A small guard time is reserved at the end of each frame to compensate for local clock drifting. The transmission starts with the time-critical packets followed by the non-time-critical packets using the assigned channel. After the LoRa end device finishes the transmissions in the current frame, it listens to CH_{ctl} and waits for the NW packet again. Figure 5.4 shows an example timeline of transmissions on different channels within two consecutive time frames. In the example, two LoRa end devices are assigned to use CH_i and CH_j, respectively. In frame n, both LoRa end devices transmit using the same SF configuration. In frame $n+1$, the LoRa end device using CH_i is assigned with a smaller SF while the other using CH_j is assigned with a larger SF. As discussed in Section 5.2, a smaller SF provides higher throughput and allows more packets to be transmitted in a frame.

In our implementation, we set the payload size of an uplink packet to 36 *bytes*, that of a NM packet to 12 *bytes*, the frame length to 5 *seconds*, and the guard time to 0.25 *seconds*.

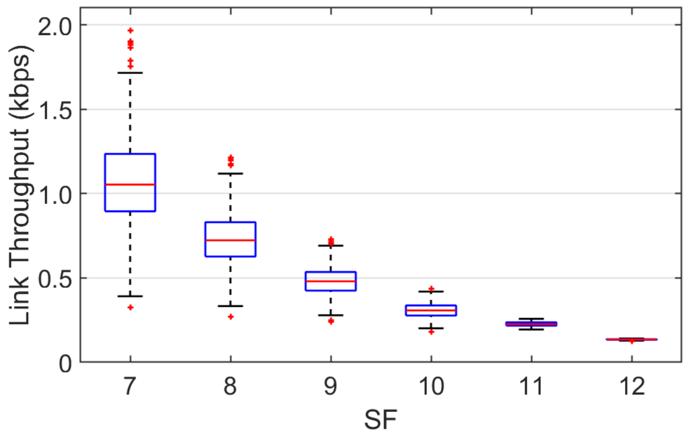
5.4 An Empirical Study on SF Configuration

To investigate the impact of SF configuration on network performance, we have performed a 14-month empirical study from March 2018 to May 2019. We configure the LoRa end devices installed on the shuttles to use all six SFs ($SF7$ to $SF12$) for packet transmissions in a round-robin fashion. In other words, a LoRa end device switches to the next SF after transmitting a packet using the current SF. The LoRa base station has collected 3.18 million uplink packets generated by six shuttles during their real-world operations (3931 *loops* in total). We have identified all lost packets and their corresponding SFs by checking the sequence IDs carried by all received packets and discarded all corrupted packets with CRC errors.

Leveraging our data trace, we first relate SF configuration to network performance. Figure 5.5 plots the box plots of the link reliability and throughput when the LoRa transmitters use different SF configurations. The link reliability in terms of PDR and the link throughput is computed every 25 *minutes*. Figure 5.5(a) and 5.5(b) clearly present the tradeoff between link reliability and throughput when the LoRa transmitters use different SF configurations. The PDR increases and the throughput decreases when the LoRa transmitters use a larger SF. Specifically, the median PDRs are 0.38, 0.43, 0.49, 0.56, 0.80, and 0.94, while the median throughput values are 1.05 *kbps*, 0.72 *kbps*, 0.48 *kbps*, 0.31 *kbps*, 0.23 *kbps*, and

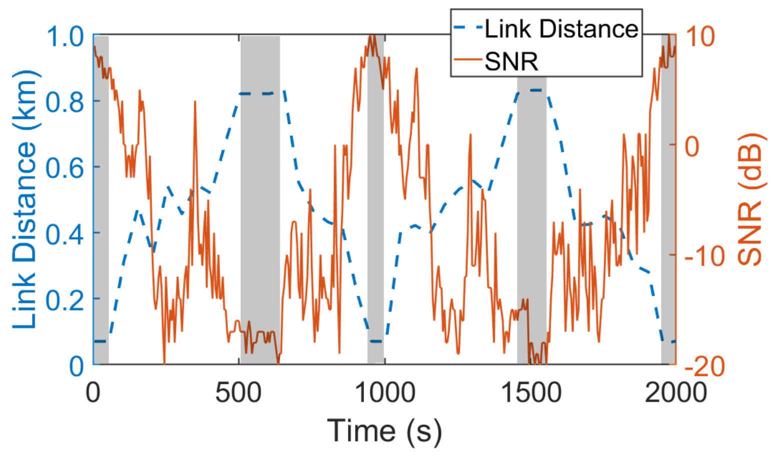


(a) Box plot of link reliability.

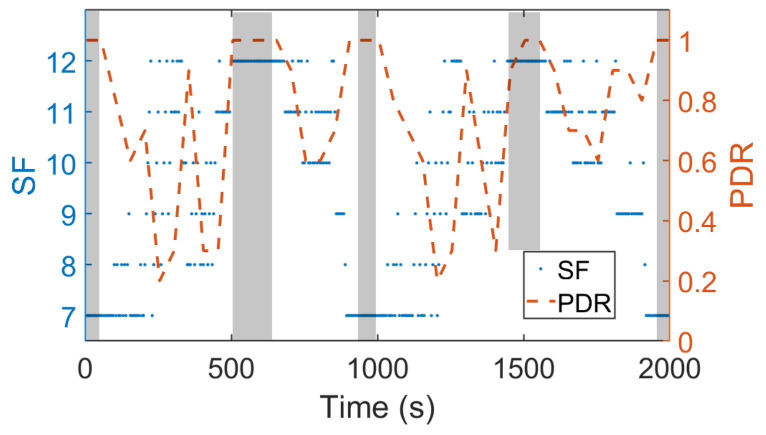


(b) Box plot of link throughput.

Figure 5.5: Link performance when the LoRa transmitters use different SF configurations.



(a) Link distance and SNR.



(b) SF selected by ADR and resulting PDR.

Figure 5.6: The link reliability changes under ADR ($window = 20, margin = 10dB$) when a shuttle circles the campus twice. The grey areas indicate the time when the shuttle stopped.

0.14 $kbps$ when the LoRa transmitters use $SF7$, $SF8$, $SF9$, $SF10$, $SF11$, and $SF12$, respectively. More importantly, the link reliability increases more significantly when the LoRa transmitters use large SFs ($SF11$ and $SF12$), while the link throughput decreases dramatically at small SFs. This indicates that increasing SF when the current SF is large may significantly enhance the link reliability at the cost of slightly reducing the link throughput, while slightly decreasing SF when the current SF is small may significantly improve the throughput without introducing too much damage on the link reliability. Those observations motivate our designs in Section 5.5.

As discussed in Section 5.2, ADR specified in LoRaWAN is designed to select SF based on the measured link quality. Unfortunately, our empirical study shows that ADR is ineffective when the LoRa end devices are in motion. For instance, Figure 5.6 shows the PDR changes when a shuttle circles around the campus twice. To understand the ineffectiveness of ADR, we install a GPS device onto that shuttle. Figure 5.6(a) plots the distance and the SNR of the link between the shuttle and the LoRa base station and Figure 5.6(b) plots the SF configurations selected by ADR and the resulting PDR over time. The shuttle made five stops around 0s, 500s, 950s, 1450s, and 1950s during that 2000s measurement. From the link distances plotted in Figure 5.6(a), we can observe that the first, third, and fifth stops are close to the LoRa base station, while the second and fourth stops are far away. As Figure 5.6(b) shows, ADR provides very good link reliability (almost 1) by selecting appropriate SFs when the shuttle was not in motion. However, the link reliability decreases significantly when the shuttle begins to move. For example, the link reliability drops to 0.21 at 255s and 0.24 at 1195s. From Figure 5.6(b), we can observe that the SF configurations selected by ADR fluctuate when the shuttle moves, resulting in unstable link reliability. There are two main reasons that cause the ADR's failure to maintain good link reliability when the LoRa end device moves quickly. First, ADR uses the maximum SNR of the last 20 SNR samples to estimate the link quality. Although it works well on stationary devices, it fails on those devices in motion because the SNR changes very fast, as Figure 5.6(a) shows. This motivates us to consider the most recent link characteristics when selecting SF (see Section 5.5). Second, ADR uses a set of theoretical SNR thresholds for SF selection rather than using the actual link reliability measured by the device.

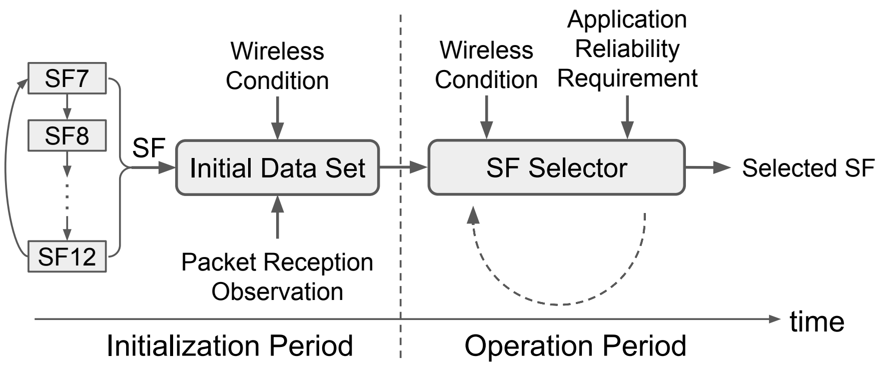


Figure 5.7: Initialization Period and Operation Period.

A recent study shows that the best-suited SNR thresholds for SF selection should be selected based on the specific physical-layer configurations of the LoRa device (e.g., packet payload length and coding rate) [91]. Moreover, we observe that using SNR measurements alone is insufficient to accurately predict the packet receptions because the receiver sensitivity of a LoRa device also depends on the RSS [92]. This motivates us to use both SNR and RSS measurements when selecting SF (see Section 5.5).

5.5 Runtime SF Control

In this section, we present the design of our runtime SF control solution that runs on the LoRa base station and selects the SF configuration for each LoRa end device to meet network performance requirements. The design goal of our solution is to maximize the data collection throughput while meeting the reliability requirement specified by the application.

5.5.1 Overview of the Solution

Our design goal is to maximize the data collection throughput while maintaining the application-specific link reliability by employing best-suited SF continuously. The SF configuration is selected based on the given link reliability requirement and link characteristics based on a machine learning model. After a LoRa end device begins to operate, our runtime SF control solution guides it through two periods: **Initialization Period** and **Operation Period**, as shown in Figure 5.7. In the Initialization Period, our runtime SF control solution controls each LoRa end device to transmit packets using all SF configurations in a round-robin fashion. It also controls the LoRa base station to measure the link characteristics when receiving every packet and observes the packet receptions under each SF configuration. This allows the LoRa base station to create the **Initial Data Set** S, in which each data element includes three pieces of information: the link characteristics (RSS, SNR, and the averaged RSS over the last 10 time frames), the SF configuration, and the packet reception result (success or failure). We empirically choose the time length of the Initialization Period that provides the best performance (see Section 5.6.1). After collecting enough data for S, our runtime SF control solution begins to periodically predict the best-suited SF configuration in the Operation Period based on a non-linear mapping between the link characteristics ($\mathbf{x}$) and the best-suited SF configuration (sf), i.e., $f : \mathbf{x} \rightarrow sf$, which is learned by our machine learning model using S. We will next present the design of our SF selector that runs in the Operation Period.

5.5.2 KNN-based SF Selector

The primary task of the SF selection algorithm is to identify the best-suited SF in each time frame as described in Section 5.5.1. We define a classification problem that periodically predicts the packet reception result (success or failure) when using each SF configuration under the given link characteristics. In each frame,

Algorithm 5: KNN-based SF Selection Algorithm

```
Input      : $\mathbf{x}$
Output     : $sf$
Data Set   : $S$
Parameters: $k$, $\mathbf{v_t}$

$d = 0$
$\mathbf{N} = [\ ]$
while $\mathbf{N}.length < k$ do
    $\mathbf{X}' = list\_link\_conditions\ (\mathbf{x}, d)$
    for $\mathbf{x}'$ in $\mathbf{X}'$ do
        $\mathbf{N}.add\ (S[x'_1][x'_2][x'_3]\ )$
    end
    $d \mathrel{+}= 1$
end
for $sf$ in $[7...11]$ do
    $v_p = 0$
    for $\mathbf{n}$ in $\mathbf{N}$ do
        $v_p \mathrel{+}= n[sf]$
    end
    if $v_p\ /\ \mathbf{N}.length > v_t[sf]$ then
        return $sf$
    end
end
$sf = 12$
```

our solution selects the smallest SF predicted with a successful packet reception to deliver the maximum throughput. While considering runtime computational efficiency, we employ the KNN algorithm [43] for classification. We adjust the parameters in the KNN algorithm at runtime to meet the reliability requirement (see Section 5.5.3). Algorithm 5 presents our KNN-based SF selection algorithm. The input of Algorithm 5 includes the link characteristics ($\mathbf{x}$) and the output is the selected SF configuration for the next frame (sf). $\mathbf{x}$ is an array of three integers (x_1, x_2, and x_3) representing the link characteristics (RSS, SNR, and averaged RSS). The Initial Data Set S is used to relate the link characteristics to the packet reception results under each SF configuration. In our implementation, S is implemented as a 3-D array that uses three array indices for the link characteristics and each array element is implemented as a linked list, which stores the observed packet reception results (1 for success and 0 for failure) using all SF configurations. The parameter k is a constant, which denotes the number of samples to be searched for in S. Based on the size and density of S, we empirically set k to 20 in our implementation. The parameter $\mathbf{v_t}$ is an array, which stores the voting thresholds for each SF. To simplify our presentation, we assume that all arrays in Algorithm 5 are free of bounds and can be accessed using any integer.

Lines 1 and 2 of Algorithm 5 initialize the distance integer d and the neighbor sample array $\mathbf{N}$. The loop from Line 3 to Line 9 searches for and stores the "nearest neighbors" in the array $\mathbf{N}$, where the nearest neighbors are the samples in S within the shortest distances to the input link characteristics ($\mathbf{x}$). We use the Euclidean distance to measure the distance between $\mathbf{x}$ and $\mathbf{x}'$:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_i (x_i - x_i')^2} \tag{5.1}$$

where x_i and x_i' ($i \in [1, 2, 3]$) are the elements of $\mathbf{x}$ and $\mathbf{x}'$. Line 4 lists all integer-valued link characteristics with the rounded distance (d) to $\mathbf{x}$ and stores them in the set $\mathbf{X}'$. The loop from Line 5 to Line 7 uses the link characteristics in each $\mathbf{x}'$ from $\mathbf{X}'$ as the indices of S and adds all samples from the indexed S elements into $\mathbf{N}$. The distance d is incremented (Line 8) until k samples have been added to $\mathbf{N}$ (Line 3). All samples with the distance d are used when a tie of distance exists. KNN uses the votes of the neighbors for classification. The loop from Line 10 to Line 18 counts the votes for each SF from $SF7$ to $SF11$ and returns the selected SF based on the voting result. For each SF configuration, v_p counts the positive votes from each neighbor $\mathbf{n}$ in $\mathbf{N}$, where a neighbor with a successful reception under the SF ($n[sf] = 1$) contributes a positive vote (Line 13). The resulting positive voting rate ($v_p/\mathbf{N}.\mathit{length}$) is compared with the voting threshold of the SF ($v_t[sf]$). If the positive voting rate is greater than the threshold, the classification result is positive for the SF (Line 15), predicting a successful packet reception. The algorithm returns with the smallest possible SF when a positive classification result is found (Line 16). The array of voting thresholds $\mathbf{v_t}$ is adjusted at runtime in response to the reliability requirement (see Section 5.5.3). If there is no positive classification result from $SF7$ to $SF11$ or the link characteristics are not available at the moment, $SF12$ is selected (Line 19) to maintain the connection.

5.5.3 KNN Voting Threshold Adjustment

The array of voting thresholds ($\mathbf{v_t}$) is a set of parameters that are critical to meet the reliability requirement specified by the application. Each element in $\mathbf{v_t}$ is a voting threshold for an individual SF because each SF may require a different threshold to meet the reliability requirement. A higher threshold requires more positive votes to predict a successful packet reception, which reduces the chance of false positive predictions.

Algorithm 6: Voting Threshold Adjustment

```
   Inputs     : $R_r$, $R_c$, $\mathbf{R_{sf}}$, $\mathbf{v_t}$
   Output     : $\mathbf{v_t}$
   Parameters: $\alpha$, $\beta$, $\gamma$
1  if $R_c < R_r$ then
2      for $sf$ $in$ [7...11] do
3          if $R_{sf}[sf] < R_r$ then
4              $v_t[sf]$ += $\alpha$
5          end
6      end
7  end
8  if $R_c > R_r + \gamma$ then
9      for $sf$ $in$ [7...11] do
10         if $R_{sf}[sf] > R_r + \gamma$ then
11             $v_t[sf]$ -= $\beta$
12         end
13     end
14 end
```

When the system begins to operate, the voting thresholds for $SF7$ to $SF11$ are initialized to 0.5. Algorithm 6 is designed to adjust the voting thresholds and is triggered when a new PDR measurement is generated or a new reliability requirement is input by the application. Given the application reliability requirement (R_r), the current link PDR (R_c), and an array of PDRs when using each SF configuration ($\mathbf{R_{sf}}$), there are two options for adjustments: (1) increasing the voting threshold for sf by α if both R_c and $R_{sf}[sf]$ are less than R_r (Line 1 to 7) and (2) decreasing the voting threshold for sf by β if both R_c and $R_{sf}[sf]$ are higher than $R_r + \gamma$ (Line 8 to 14). Algorithm 6 keeps the previous voting thresholds if no change is needed. In our implementation, we set α to 0.1, β to 0.05 which allows a fast response when the actual reliability fails to meet the requirement and a slow adjustment when the actual reliability is high. While increasing the voting thresholds helps increase the link reliability, decreasing the voting thresholds is designed to increase the link throughput because the selector has a greater chance to select smaller SFs with higher data rates. The value of γ is set to 0.05 in our implementation which aims to maintain the link reliability between R_r and $R_r + 0.05$.

5.6 Evaluation

To validate the efficiency of our runtime SF control solution in maximizing the link throughout while meeting the application-specified reliability requirement, we performed a series of experiments. We first empirically identify the best-suited length of the Initialization Period. We then examine whether our solution can consistently meet the reliability requirement specified by the application. We evaluate our solution's effect on increasing the data collection throughput, and compare its performance against three baselines. Finally, we evaluate the runtime efficiency of our solution by measuring its execution time on a Raspberry Pi computer. To ensure a fair comparison between our solution and baselines, we apply all solutions on the same data trace collected from our 14-month empirical study with six shuttles (see Section 5.4). We also identify the optimal SF selections by analyzing the

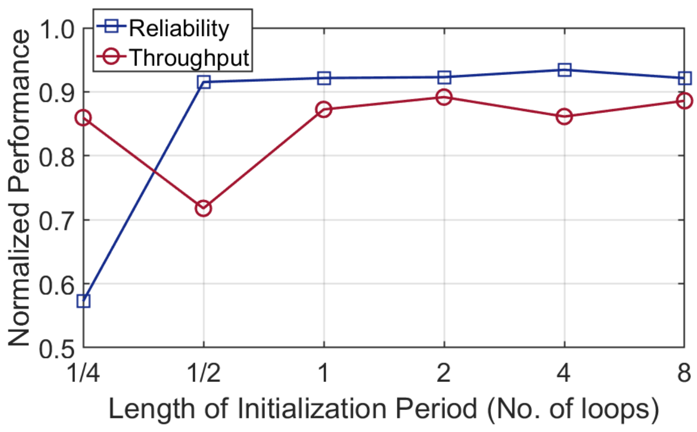
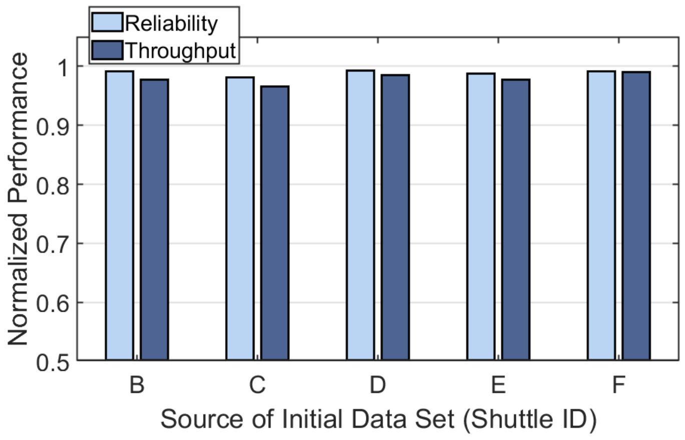


Figure 5.8: Performance when using the Initial Data Set with different sizes. Performance is normalized to the one using the optimal selections.

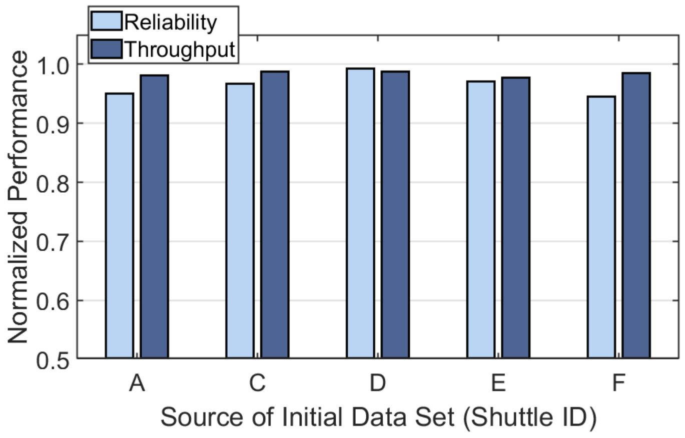
entire data trace. Please note that the optimal solution cannot be implemented at runtime and is only for comparison purposes.

5.6.1 Impact of Initialization Period Length

As discussed in Section 5.5.1, our runtime SF control solution first controls the LoRa end devices installed on the shuttles to use all six SFs ($SF7$ to $SF12$) for packet transmissions in a round-robin fashion to generate the Initial Data Set S. The network may experience poor reliability and low throughput in the Initialization Period. Thus, it is beneficial to keep the Initialization Period as short as possible. We run experiments to study the impact of the length of the Initialization Period on network performance. Figure 5.8 shows the network reliability and throughput when our runtime SF control solution uses S with different sizes. All results are normalized to those using the optimal selections. As Figure 5.8 shows, the normalized reliability is very low (0.57) when S has the collected data when a shuttle traveled the first quarter of its route. After the shuttle traveled the first half of its route, the normalized reliability increases to 0.92, but the normalized throughput is only 0.72. The normalized throughput increases to 0.87 and



(a) Using the initial data collected from different shuttles for Shuttle A.



(b) Using the initial data collected from different shuttles for Shuttle B.

Figure 5.9: Performance when using the Initial Data Set collected from one shuttle on another. Performance is normalized to the one when using the initial data collected from the same shuttle. The Initial Data Set includes one loop of data. $R_r = 0.8$.

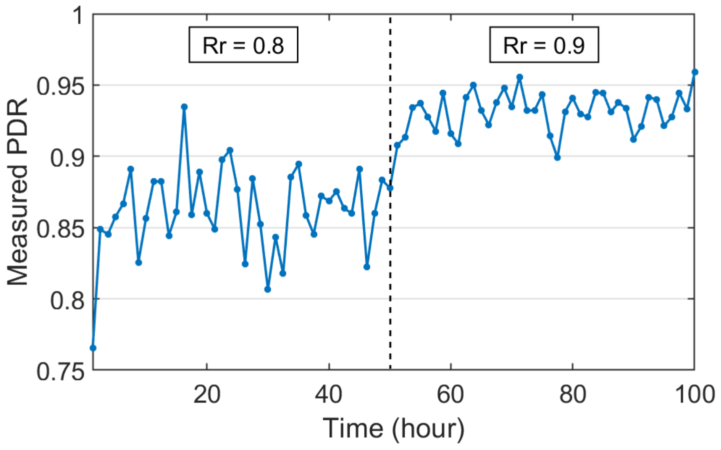
the normalized reliability is 0.92 when the shuttle traveled the whole route once. More data in S does not provide much help on improving the throughput and reliability. The results show that collecting one loop of data to create the Initial Data Set is enough for our KNN-based SF selector to provide good SF selections at runtime.

5.6.2 Sharing the Initial Data among Shuttles

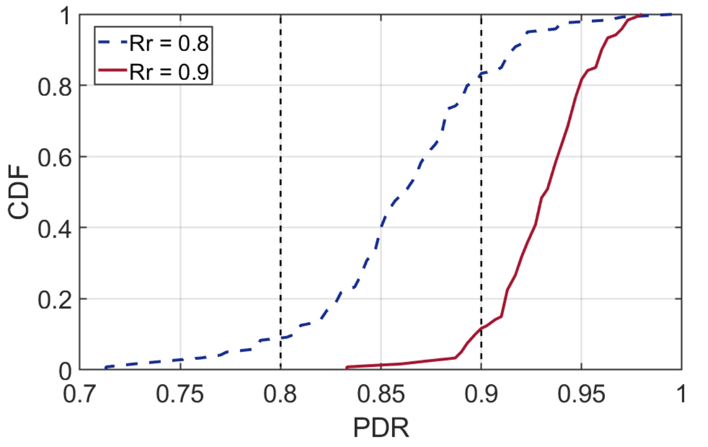
We run experiments to explore the feasibility of sharing the Initial Data Set collected from one shuttle with other shuttles. Figure 5.9 plots the reliability and throughput performance when using the initial data collected from other shuttles on Shuttles A and B. The results are normalized to the one when using the initial data collected from the same shuttle. As Figure 5.9(a) shows, the normalized reliability ranges from 0.98 to 0.99 when using the initial data collected from Shuttle B, C, D, E, and F for Shuttle A, while the normalized throughput ranges from 0.96 to 0.99. Similarly, the normalized reliability and throughput are not less than 0.94 and 0.97, respectively, when using the initial data collected from different shuttles for Shuttle B. The absolute reliability is not less than 0.87 and 0.83 when using the initial data collected from different shuttles on Shuttles A and B, respectively, which meets the reliability requirement specified by the application ($R_r = 0.8$). The results show that using the Initial Data Set collected from one shuttle for other shuttles only slightly degrades the performance of our runtime SF control solution when the shuttles follow the same route; therefore it is feasible to share the Initial Data Set among different shuttles, which significantly reduces the initialization overhead.

5.6.3 Effectiveness of our Runtime SF Control Solution

We perform a series of experiments to examine whether our runtime SF control solution can consistently meet the reliability requirement specified by the application. We configure the application to input different reliability (PDR) requirements and measure the actual PDRs at the LoRa base station. Figure 5.10(a)



(a) An example data trace of PDR measurements. The application changes its PDR requirement from 0.8 to 0.9 at the 51st hour. The PDR measurement is computed in every 1.25 hours.

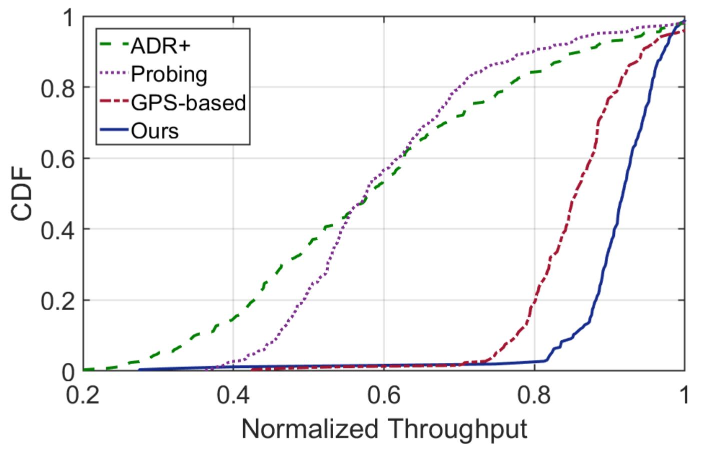


(b) CDF of PDR measurements.

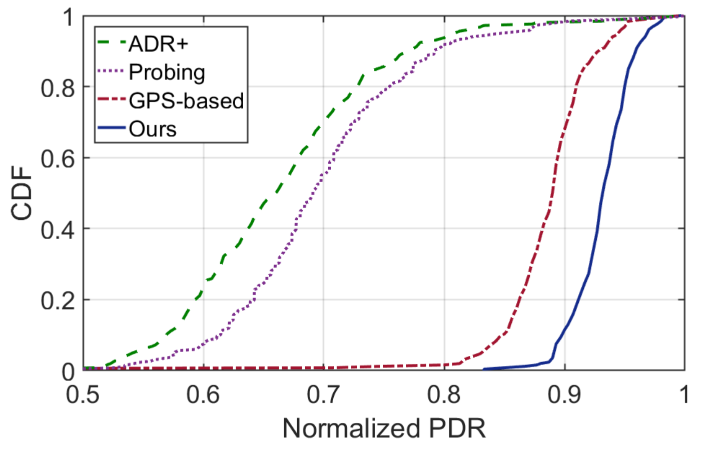
Figure 5.10: Performance under different application reliability requirements.

plots the example PDR measurements collected from a shuttle for more than 100 *hours*. In this example, the application inputs 0.8 as the PDR requirement at the beginning of the Operation Period and then changes the requirement to 0.9 at the 51st hour. As Figure 5.10(a) shows, our runtime SF control solution can always meet the application reliability requirement except for the first measurement. The slightly lower reliability (0.76) in the first measurement is caused by the voting threshold adjustments performed at the beginning of the Operation Period. More importantly, our runtime SF control solution only takes 117 μs to select a new SF to accommodate the reliability requirement changes issued by the application. This demonstrates the time efficiency of our SF selections. Figure 5.10(b) plots the Cumulative Distribution Function (CDF) of PDR measurements under different application reliability requirements. The PDR measurement is computed every 25 *minutes*. In more than 95.8% and 90.2% of the time, our runtime SF control solution provides good SF selections, which successfully meet the application reliability requirements, 0.8 and 0.9, respectively.

Our runtime SF control solution is designed to maximize the link throughput. We compare the throughput provided by our solution against three baselines: ADR+, Probing, and GPS-based. ADR+ is an enhanced version of ADR, which takes input from the average (instead of maximum) SNR of the last 20 packets and selects SFs based on the required SNR for each SF configuration [38]. Probing is a LoRa transmission parameter selection algorithm based on the measured link Packet Reception Ratio (PRR) [39]. GPS-based is a baseline that we create by installing GPS devices on the shuttles and using GPS coordinates to select SF configurations. Figure 5.11 plots the comparisons among four solutions, where all



(a) CDF of normalized throughput.



(b) CDF of normalized PDR.

Figure 5.11: Performance comparisons between our solution and three baselines. Performance is normalized to the one using the optimal selections.

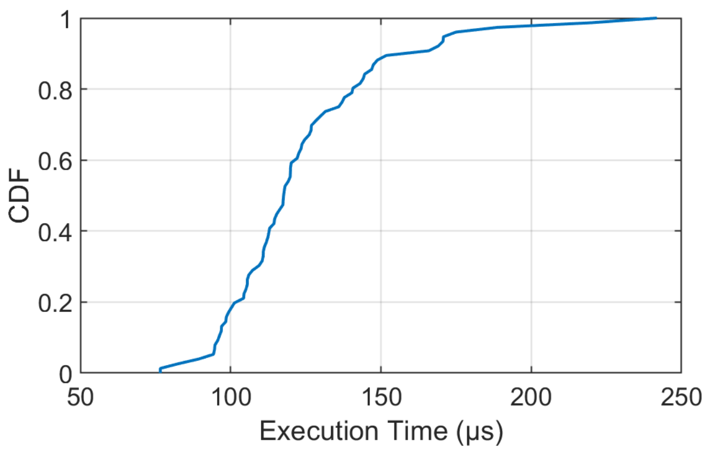


Figure 5.12: The execution time of SF selections on a Raspberry Pi computer.

results are normalized to the optimal values. Figure 5.11(a) shows the CDF of normalized throughput. The median throughput normalized to optimal is 0.58, 0.57, 0.86, and 0.92, when the LoRa base station runs ADR+, Probing, GPS-based, and our solution, respectively. Figure 5.11(b) shows the CDF of normalized PDR. The median normalized PDR is 0.66, 0.69, 0.89, and 0.93 when the LoRa base station runs ADR+, Probing, GPS-based, and our solution, respectively. As Figure 5.11(a) and 5.11(b) show, our runtime SF control solution consistently provides the highest throughput and best reliability among all solutions. The result also indicates that the measured link characteristics can be used reliably to select good SF configurations and there is no need to install additional GPS devices which are cost and time inefficient.

5.6.4 Time Efficiency of our Runtime SF Control Solution

Our KNN-based SF selection is designed to be lightweight. We measure the time duration taken by our KNN-based SF selector to select the best-suited SF configuration. We record the time of the events when the input is fed into the selector and the output (i.e., SF configuration) is generated. For this experiment, we repeat the measurement 50,000 times on the Raspberry Pi computer with a 900MHz quad-core ARM CPU and the RAM of 1GB. Figure 5.12 plots the CDF of the time duration of each run. As Figure 5.12 shows, the median execution time is 117 μs. 90% and 99% of the SF selections finish within 165 μs and 241 μs, respectively. These results demonstrate the good time efficiency of our SF selector as well as the advantage of using the KNN algorithm.

5.7 Conclusion

Satellite and cellular technologies are traditionally used to collect real-time data from running vehicles to the base station through their long-distance links. However, such systems are often costly because of their use of expensive devices and licensed frequency bands, which prevents them from being used in many application scenarios. As an emerging LPWAN technology, LoRa has been used as a low-cost alternative that provides capability for long-range data collection to low data rate applications. In this chapter, we present a low-cost LoRa-based wireless network, ShuttleNet, that collects real-time data from six shuttles circling our university campus and has operated in the real world for more than a year. When implementing ShuttleNet, we find that the selection of LoRa SF poses a significant challenge because of its effects on two conflicting QoS metrics. To investigate the impact of SF configuration on network performance, we have performed a 14-month empirical study. Our empirical study shows that a larger SF provides higher network reliability at the cost of lower throughput. More importantly, we observe that the link reliability increases more significantly when the LoRa transmitters use large SFs ($SF11$ and $SF12$), while the link throughput decreases dramatically at small SFs. This indicates that increasing SF when the current SF is large may significantly enhance the link reliability at the cost of slightly reducing the link throughput, while slightly decreasing SF when the current SF is small may significantly improve the throughput without introducing too much damage on the link reliability. Those observations motivate us to develop a runtime SF control solution that employs the KNN algorithm to adapt the SF configuration based on the link characteristics. We compare our solution against three state-of-the-art baselines and observe that ours consistently provides the highest throughput and the best reliability among all solutions.

6 DIME: Direct Message Dissemination for Industrial Wireless Networks via Cross-Technology Communication

6.1 Introduction

Over the past decade, industrial wireless networks have been widely adopted in industrial facilities to connect sensors, actuators, and controllers. Recent advancements in radio hardware and embedded platforms have provided battery-powered wireless modules which are readily available and cost-efficient to build industrial wireless networks by easily and inexpensively retrofitting existing sensors and actuators without the need to run cables for communication and power. Research efforts in recent years have proposed autonomous scheduling [51, 52] and self-configuration [93] solutions for wireless mesh networks based on the IEEE 802.15.4 standard, enabling flexible and scalable deployments of industrial wireless networks.

Industrial wireless networks have critical demands for reliable and real-time communication. Failing to achieve those demands may result in degraded productivity, extra maintenance cost, or safety hazard. However, those networks often suffer high network management complexity due to a large number of network nodes and a mesh network topology, posing a significant challenge to achieve those critical demands. Deliveries of critical network management data, such as time information, control commands, and urgent alerts, may suffer insufficient reliability and undesirable latency due to hop-by-hop transports. As a result, the network may take a long time to stabilize the performance at startup and experience considerable latency for data deliveries in feedback loops.

Emerging Low-Power Wide-Area Network (LPWAN) technologies, such as LoRa [94], enable one-hop long-distance links as an alternative to multi-hop links, offering new opportunities to address the high management complexity of industrial wireless networks [21]. However, adding new radio modules may increase the cost and complexity to deploy and maintain the system. Fortunately, new Cross-Technology Communication (CTC) techniques have enabled direct messaging from a LPWAN radio to 802.15.4-based field devices without any hardware change on the field devices [54, 55]. Using CTC techniques is therefore a promising solution to overcome the high management complexity of industrial wireless networks by introducing long-distance links.

In this chapter, we develop the Direct Message Dissemination (DIME) system that leverages the CTC technique to efficiently time synchronize the wireless devices in the network and enable direct message deliveries through long-distance links instead of hop-by-hop transports. Specifically, we make the following contributions:

- We extend TSCH and propose DIME-MAC, which is a MAC protocol that enables direct message dissemination via CTC-based long-distance links;
- We design a scheduling solution that supports direct time synchronization and direct message deliveries for feedback loops in industrial wireless networks;

- We deploy DIME on our testbed and reduce the network management complexity of 40 field devices without any hardware modification on the field devices;
- Our testbed experiments show that DIME significantly improves the end-to-end reliability, reduces the end-to-end latency, and quickly stabilize the network performance at startup compared to the state of art.

The remainder of the chapter is organized as follows. Section 6.2 introduces the background of CTC, TSCH, and transmission scheduling. Section 6.3 present the design of DIME-MAC and our scheduling solution. Section 6.4 shows the testbed deployment and evaluation of DIME. Section 6.5 concludes the chapter.

6.2 Background

DIME employs the following technologies and techniques to reduce the management complexity of industrial wireless networks.

CTC. Recent research efforts have demonstrated various CTC techniques that support direct message exchange among heterogeneous wireless technologies (e.g., WiFi, ZigBee, Bluetooth, and LoRa) operating under the same frequency band. CTC techniques not only promote the coexistence of heterogeneous wireless technologies but also expand the features of existing wireless technologies. For instance, the communication range of ZigBee radios can be extended with the CTC technique that enables direct messaging from LoRa to ZigBee. Shi et al. have demonstrated that all ZigBee devices on a testbed can detect transmissions from a single LoRa radio and decode data from the received signal strength (RSS) patterns, significantly extending the communication range covered by a single-hop link [55]. We employ this unique feature of direct messaging from LoRa to ZigBee to support time synchronization and downlink message deliveries over a large area.

TSCH and Transmission Scheduling. TSCH [50] is a MAC protocol standardized in IEEE 802.15.4e to provide time-deterministic packet deliveries for industrial process control and automation. TSCH divides time into timeslots (or slots) and enables channel hopping in every timeslot. Under TSCH, all network nodes need to keep globally time synchronized to support time slotted access. Starting from the coordinator node, each synchronized node shares its time information to its neighbors by periodically broadcasting Enhanced Beacon (EB). Contained in EB, the Absolute Slot Number (ASN) indicates the network time in the term of timeslots. ASN is set to zero when a network starts and increased by one at the end of each timeslot. Timeslots are grouped into slotframes which appear periodically every L_{SF} timeslots, where L_{SF} is called slotframe length. A timeslot in a slotframe is specified by the time offset (t_o), the channel offset (c_o), and the type of operation (e.g., transmission, reception, or sleep). The time offset (t_o) of a timeslot is represented by the modulo operation between ASN and L_{SF}:

$$t_o = mod(ASN, L_{SF}) \tag{6.1}$$

A sequence of channels used for channel hopping, called Frequency Hopping Sequence (FHS), is known by all nodes. The frequency channel used in a timeslot is determined by FHS, ASN, the channel offset (c_o), and the length of FHS (L_{FHS}):

$$channel = FHS(mod(ASN + c_o, L_{FHS})) \tag{6.2}$$

A TSCH schedule determines a node's operation type and channel offset in each timeslot. Recent research efforts have proposed autonomous scheduling schemes (such as Orchestra [51] and ALICE [52]), where each node maintains its schedule automatically based on its local state of the routing protocol, without the need of a centralized scheduler or inter-node schedule negotiation. Although using autonomous scheduling in TSCH can enhance the scalability of the network, there still exists the high complexity issue of network management. Deliveries of critical network management data, such as time information, control commands, and urgent alerts, may suffer insufficient reliability and undesirable latency due to hop-by-hop transports. To solve this issue, we leverage the CTC technique to extend the TSCH MAC protocol and improve the reliability and latency performance in autonomously scheduled networks by reducing the network management complexity.

6.3 Design of DIME

6.3.1 System Overview

The goal of DIME is to enable the direct message dissemination from the root node to the field nodes in mesh wireless networks for synchronization and downlink messaging as an efficient alternative to hop-by-hop transports. In the design of DIME, the application data is transferred through one or more data flows in the network, where each data flow contains a feedback loop from the source node to the destination node. The source node generates the application packets, which are forwarded by the the relay nodes to the root node using the 802.15.4-based multi-hop link. The root node dispatches the feedback packets to the destination

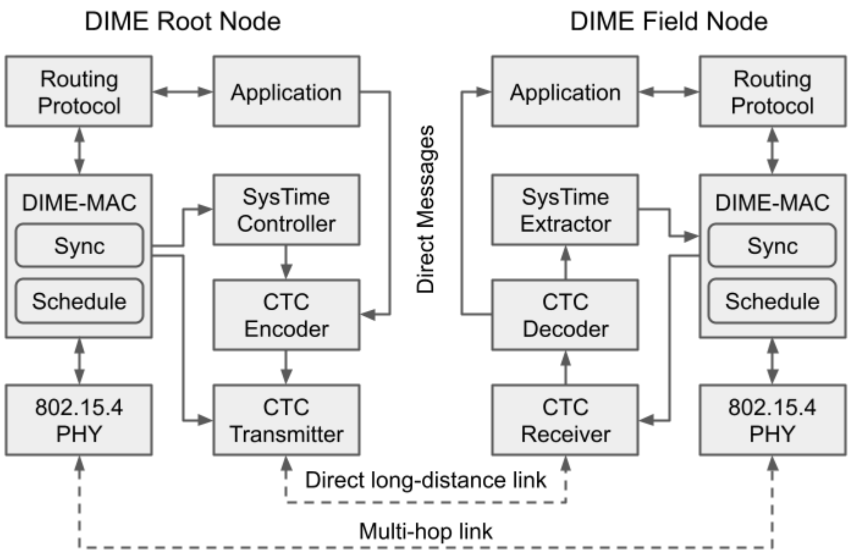


Figure 6.1: Software architecture of DIME.

node via both the CTC-based direct link and the 802.15.4-based multi-hop link. Figure 6.1 describes the software architecture of DIME on the root node and the field nodes. On both types of nodes, DIME-MAC is the MAC layer protocol that keeps the synchronization of the network and maintains the time schedule for each node to transmit and receive. Also, a routing protocol based on RPL maintains a routing table on each node and forwards packets between the application layer and the MAC layer. Furthermore, there are components designed specifically for the root node and the field nodes.

The root node. The root node is equipped with two radios of different types: the main 802.15.4 radio and a secondary radio (LoRa in our deployment), where the main radio features the 802.15.4 physical layer while the secondary radio functions as a CTC transmitter to support the direct long-distance links from the root node to the field nodes. The SysTime Controller is periodically activated by DIME-MAC to update the current system time. The application layer generates Direct Messages for downlink deliveries. The CTC encoder encodes the system time and

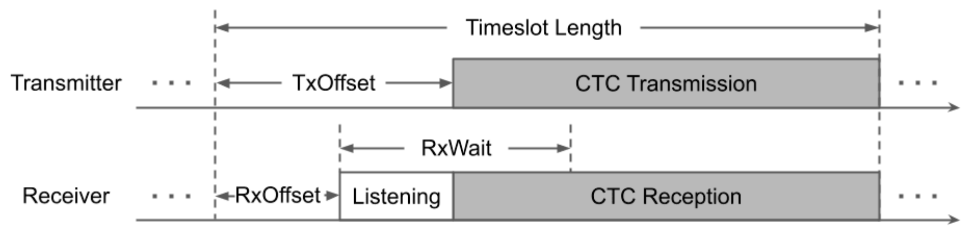


Figure 6.2: The structure of a CTC timeslot.

Direct Messages into CTC transmission patterns. The CTC transmitter transmits the encoded system time and Direct Messages at dedicated timeslots under the transmission control of DIME-MAC.

The field nodes. Each field node is equipped with a single 802.15.4 radio, which not only features the 802.15.4 physical layer but also functions as the CTC Receiver. The CTC Receiver is activated by DIME-MAC at dedicated timeslots to detect CTC transmissions, which is then decoded by the CTC Decoder. The SysTime Extractor obtains the current system time from the decoded message and input the time information to DIME-MAC. The decoded Direct Messages are fed into the application layer.

6.3.2 DIME-MAC

DIME uses a MAC protocol named DIME-MAC which is extended from TSCH. DIME-MAC inherits timeslots and slotframes from TSCH, where timeslots are grouped into slotframes that repeat over time.

CTC Timeslots. DIME-MAC introduces a new type of timeslots called CTC timeslots, which are dedicated for direct message deliveries via CTC-based long-distance links. The unique feature of CTC timeslots is that the end of each timeslot is marked by the end of the CTC transmission, which can be used for direct and accurate time synchronization. The structure of CTC timeslots is defined in

Figure 6.2, where TxOffset, RxOffset, and RxWait are three timing parameters inherited from TSCH. The CTC transmission starts at TxOffset, which is a time offset determined by the time duration of the CTC transmission to make sure that the CTC transmission ends at the end of the timeslot. TxOffset should be no less than the recommended value in TSCH (e.g., $3ms$ for $15ms$ timeslots) to make CTC transmissions tolerable to the clock drifts on field devices. The receiver starts listening to the channel at RxOffset and keeps listening for the start of a valid CTC message for the duration of RxWait. If the receiver detects no CTC message started in this duration, it skips the reception and goes to sleep; otherwise, it continues the reception till the end of the CTC transmission. There is no acknowledgement performed in CTC timeslots because they are designed to broadcast. The CTC coding space, i.e., the set of the different values that can be encoded in a CTC timeslot, is limited by the CTC physical layer and the timeslot length. The coding space can be divided into subsets for different purposes, such as time synchronization and application messages.

Direct Synchronization. As discussed in Section 6.2, TSCH uses EB for time synchronization. DIME-MAC replaces EB with Direct Beacon (DB) to synchronize the network directly and allow new nodes to join the network. Similar to EB, DB also contains time information and is broadcasted periodically. Unlike EB is broadcasted by all nodes in the network, DB is only transmitted by the root node via CTC-based long-distance links, reducing the network management complexity for time synchronization. In DIME-MAC, all field nodes consider the root node as their time source and periodically listen to DB to keep synchronized. Each DB transmission is encapsulated in a CTC timeslot. When a DB transmission is finished, all field nodes use the ending time of that timeslot to synchronize their local clocks. DB carries the current Synchronization Period Number (SPN) to indicate the absolute time since the network starts. SPN is initialized as 0 and incremented by 1 after each time it is transmitted. SPN can be encoded in two or more consecutive DB transmissions, where the last DB transmission contains a special label that indicates the end of SPN. Assume there are S_{DB} possible values in DB's coding space, each SPN is encoded in N_{DB} DB transmissions, and the period of DB transmissions is T_{DB}. Then, the maximum time T_{max} that can be represented by SPN using DB is

$$T_{max} = T_{DB} \times N_{DB} \times (S_{DB} - 1)^{N_{DB}-1} \tag{6.3}$$

For example, if $S_{DB} = 8$, $N_{DB} = 10$, and $T_{DB} = 6\ s$, then T_{max} is more than 76 years. Based on the received SPN, field nodes can calculate the current ASN by

$$ASN = SPN \times T_{DB} \times N_{DB} \tag{6.4}$$

Joining the Network. There are two ways for a new node to join the network: joining at the network initialization or when the network is operating. At the network initialization, the CTC transmitter continuously broadcasts beacons with a constant pattern for a short period of time (e.g., 30 s) before the network starts to operate, allowing all field nodes to adjust their CTC physical layer parameters (e.g., the RSSI threshold) based on the link measurements in this period. If a new node joins when the network is operating, it uses the default CTC physical layer parameter and listens for DB transmissions to obtain a complete SPN and calculate the current ASN based on the SPN. When joining at network initialization, there is no need to obtain the SPN since it is known to start from 0.

6.3.3 Scheduling of DIME

The goal of scheduling is to determine each node's operation type and channel offset in each timeslot. DIME adopts the scheduling scheme of Orchestra with the properties of timeslots and slotframes. DIME schedules timeslots based on three types of concurrent slotframes used for synchronization, routing, and application traffic, respectively. The schedules of all slotframes are combined into a single schedule at runtime. Each slotframe has a priority level that determines whether to use or yield a timeslot when encountering a conflict during the combination with the other slotframes. The lengths of all slotframes are mutually prime to minimize the schedule conflicts.

DB Slotframe: The slotframe with the highest priority is dedicated to DB transmissions for time synchronization. The only active timeslot scheduled per DB slotframe is a CTC slot for DB transmission, where the time offset (t_o) is 0 and the operation channel is fixed to the best one (we choose channel 26). The root node broadcasts DB and every field node listens to DB in this CTC slot. We use $L_{SF}^{DB} = 397$ slots as the default length of DB slotframes.

RPL Slotframe: The slotframe for RPL routing has the second priority. One common shared (CS) timeslot is scheduled per RPL slotframe for RPL broadcast messages, where the time offset (t_o) is 0 and the channel offset (c_o) is 1. We use 31 slots as the default length of RPL slotframes.

Application Slotframe: The slotframe for application traffic has the lowest priority among the three slotframe types. The application slotframe has two possible phases – the uplink phase followed by the optional downlink phase. The slotframe can be configured with the uplink phase alone if the downlink phase is not required. The uplink phase assigns every field node with a sender-based dedicated (SBD) slot to collect application data (e.g., sensing data) from the source nodes to the root node, while the the downlink phase assigns every field node with a receiver-based dedicated (RBD) slots to deliver application feedback data (e.g., actuation commands) from the root node to the destination nodes. In the uplink phase, every node listens to its child nodes (if any) and transmits uplink data (if any) in its SBD slot. The time offset of each SBD slot (t_o^{SBD}) is calculated by the modulo operation between the sender node ID ($txID$) and the length of the uplink phase (L_{UP}):

$$t_o^{SBD}(txID) = mod(txID, L_{UP}) \tag{6.5}$$

Based on Eq. 6.5, each node calculates its uplink transmitting slot using its own node ID and calculates its uplink receiving slot(s) using its child node ID(s). After the uplink phase, the downlink phase begins with a CTC slot, in which the root node groups the downlink data for all destination nodes into a CTC transmission. [1] The time offset of the CTC slot (t_o^{CTC}) equals L_{UP} since it follows the uplink phase. All destination nodes listen to that CTC slot and decode their data if successfully received. After the CTC slot, every node on each downlink path (the path from the root node to the destination node) listens in its RBD slot and transmits downlink data (if any) to its child node on the path until the destination node is reached. The time offset of each RBD slot (t_o^{RBD}) is calculated by the receiver node ID

[1]We assume that the downlink data for all destination nodes can be grouped into one CTC slot because of the small size of typical downlink application data (such as actuation commands).

$(rxID)$, L_{DP}, and the length of the downlink phase (L_{DP}):

$$t_o^{RBD}(rxID) = mod(rxID, L_{DP} - 1) + L_{UP} + 1 \tag{6.6}$$

Based on Eq. 6.6, each node on the downlink paths calculates its downlink receiving slot using its own node ID and calculates its downlink transmitting slot(s) using its child node ID(s) on all downlink paths. The channel offset (c_o) of the application slotframe is fixed to 2. To ensure contention-free transmissions in both uplink and downlink phases, both L_{UP} and L_{DP} should be greater than the maximum node ID in the network. We choose $L_{UP} = 50$, $L_{DP} = 51$, and $L_{SF}^{APP} = L_{UP} + L_{DP} = 101$ as the default lengths in our deployment with 40 field nodes.

6.4 Evaluation

In this section, we present extensive experiments on our testbed with DIME and Orchestra to demonstrate the unique benefits of DIME in reducing the network management complexity and improving the network performance under different data rates.

6.4.1 Experimental Setup

We run experiments on our testbed consisting of 40 field devices and a root device in an office environment. Each field device is a TelosB mote and the root device is a TelosB motes bundled with a 2.4 GHz LoRa module. We configure 8 data flows on our testbed, where each data flow configured with a source node and a destination node. The relay nodes forward packets on the uplink and downlink

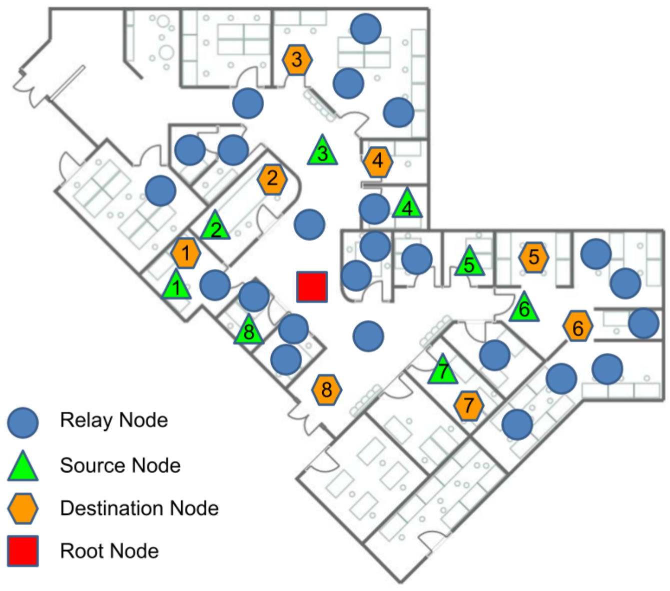


Figure 6.3: Testbed deployment with 8 data flows. The data flow numbers are marked at the source and destination nodes.

paths, where the uplink paths are configured by RPL at runtime and the downlink paths are preset as static ones due to the memory limit of TelosB motes. Figure 6.3 shows the device deployment on our testbed with different types of nodes on each data flow.

We configure DIME and Orchestra with four concurrent slotframes – one synchronization, one routing, and two application slotframes. We use the default slotframe lengths and settings for the synchronization and routing slotframes in both DIME and Orchestra. The slotframe lengths of the two application slotframes are set to 101 and 103 slots. Among the two application slotframes, the slotframe with 101 slots has a higher priority and is configured with the default uplink and downlink phases, while the slotframe with 103 slots has a lower priority and is configured with the uplink phase only. Orchestra is configured to use the same schedule of the application slotframes as DIME but skip the CTC timeslot in the downlink phase. In our deployment, the length of a timeslot is set to $15ms$.

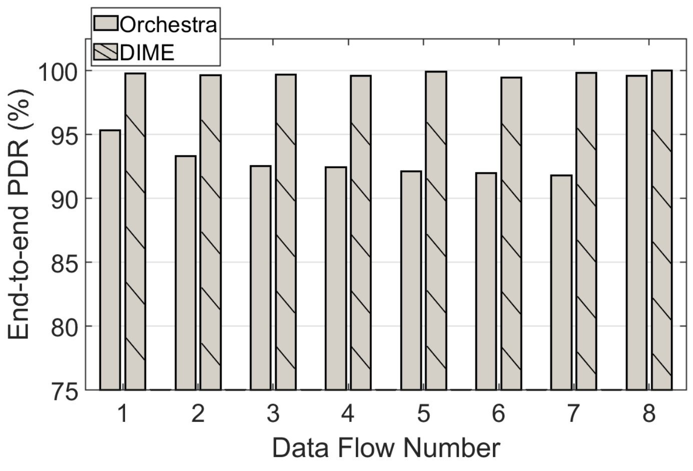
We configure the experiments as a real-time sensing-actuating application, where each source node generates an uplink packet at a random time within the data generation period that repeats over time. After the root node receives an uplink packet, it generates a downlink packet to be delivered to the destination node of the same data flow.

The evaluated performance metrics include end-to-end packet delivery ratio (PDR), end-to-end latency, and radio duty cycle. For a given data flow, the end-to-end PDR is the ratio of the number of packets received by the destination node to the number of packets transmitted by the source node. The end-to-end latency is the time duration between the uplink packet transmission at the source node and the downlink packet reception at the destination node. The radio duty cycle is the percentage of timeslots in which the radio is on, indicating the radio energy consumption.

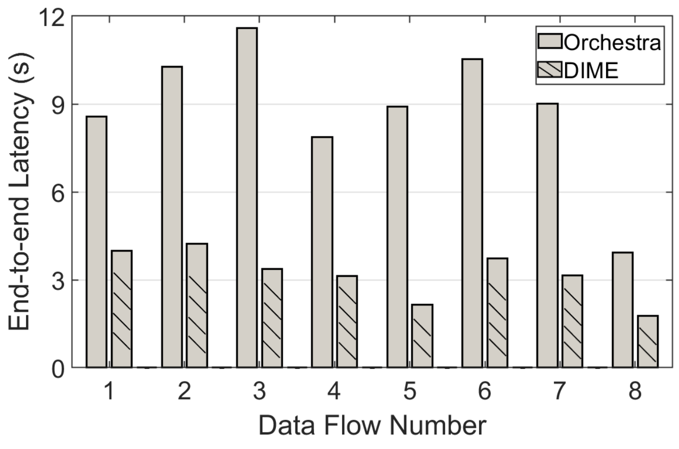
6.4.2 Performance at High Data Rate

We first run DIME and Orchestra for 24 hours to measure end-to-end PDR and end-to-end latency under a high data rate setting, where the data generation period is $10s$ at the source nodes. Figure 6.4 and Figure 6.5 summarize our results.

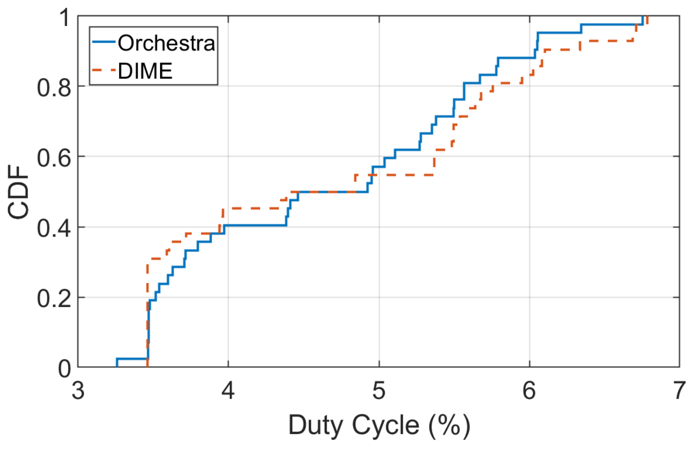
Figure 6.4(a) shows the end-to-end PDR of each data flow, where DIME achieves the PDR above 99.5% in all data flows, outperforming Orchestra which has the lowest PDR of 91.8% at data flow 7. Figure 6.4(b) shows the average end-to-end latency of each data flow, where DIME achieves the latency no higher than 4.23 seconds and significantly reduces the latency compared to Orchestra. For instance, the average end-to-end latency of data flow 3 is 3.36 seconds under DIME and 11.58 seconds under Orchestra. Figure 6.4(c) plots the CDF of



(a) End-to-end PDR.



(b) End-to-end latency.



(c) Radio duty cycle of field nodes.

Figure 6.4: Performance comparison under the high data rate setting.

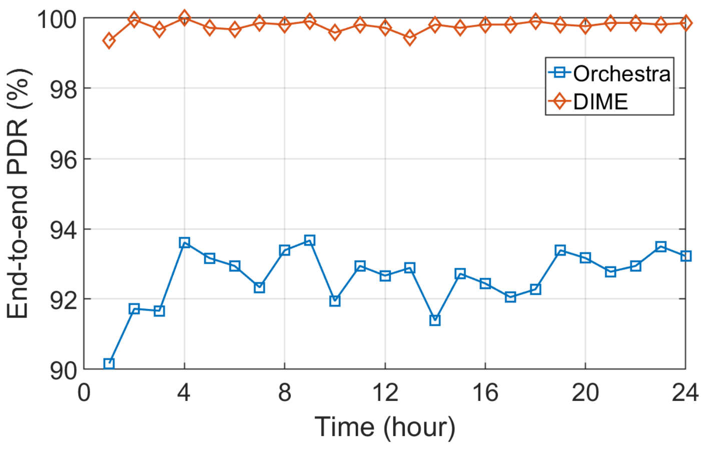
the radio duty cycle of all field nodes under DIME and Orchestra, where the two approaches deliver similar performance. The radio duty cycle ranges from 3.46% to 6.78% under DIME and ranges from 3.26% to 6.75% under Orchestra. The median radio duty cycle is 4.61% under DIME and 4.69% under Orchestra. These results clearly demonstrate the benefits of DIME on improving the reliability and reducing the latency without the overhead of increased radio duty cycle.

Figure 6.5 compares the performance of the entire network under DIME and Orchestra over 24 hours, where each data point shows the performance within the duration of one hour. As Figure 6.5(a) shows, the end-to-end PDR keeps above 99.3% under DIME and fluctuates between 90.2% and 93.7% under Orchestra. As Figure 6.5(b) shows, the end-to-end latency keeps less than 3.61s under DIME and varies between 8.23s and 10.54s under Orchestra. On average, DIME improves the end-to-end PDR from 92.6% to 99.8% and reduces the end-to-end latency from 9.03s to 3.06s compared to Orchestra. These results show that DIME constantly outperforms Orchestra over a long time of operation at the high data rate.

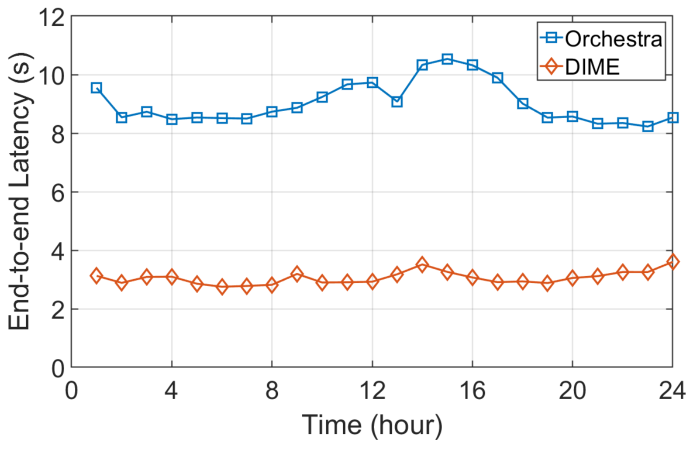
6.4.3 Performance at Low Data Rate

We also run DIME and Orchestra for 24 hours under a low data rate setting, where the data generation period is 60s. Figure 6.6 and Figure 6.7 summarize our results.

Figure 6.6(a) shows the end-to-end PDR of each data flow, where DIME and Orchestra achieve the PDR above 99.93% and 99.52%, respectively. Figure 6.6(b) shows the average end-to-end latency of each data flow, where DIME reduces the latency by 6.53s on average compared to Orchestra. For instance, the end-to-end latency of data flow 6 is 1.57s under DIME and 11.42s under Orchestra.

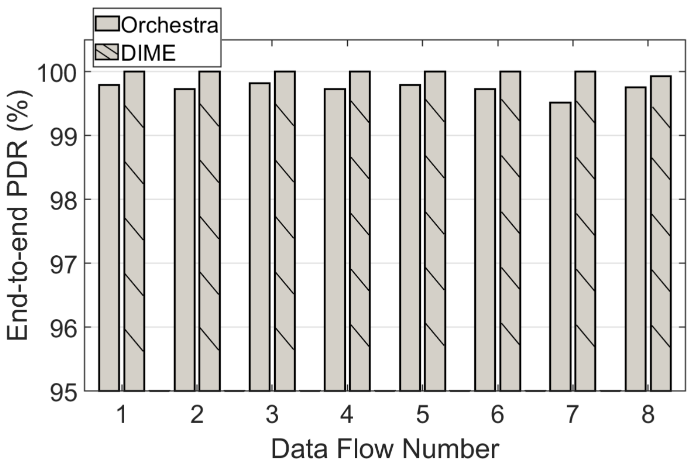


(a) End-to-end PDR.

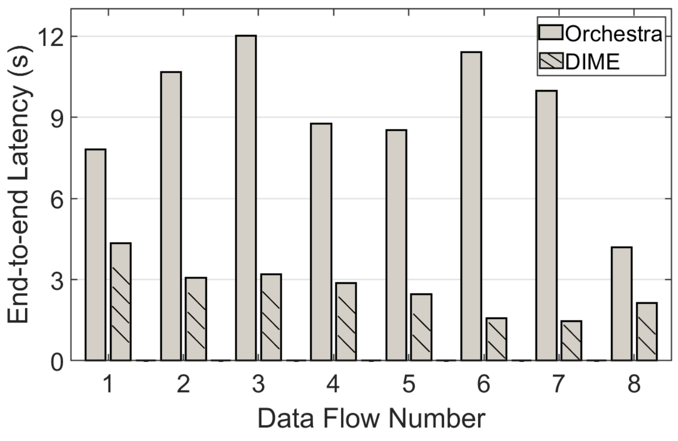


(b) End-to-end latency.

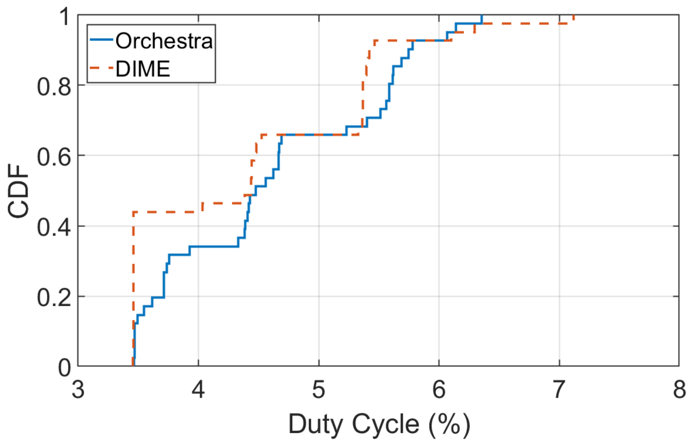
Figure 6.5: Network performance over 24 hours under the high data rate setting.



(a) End-to-end PDR.



(b) End-to-end latency.



(c) Radio duty cycle of field nodes.

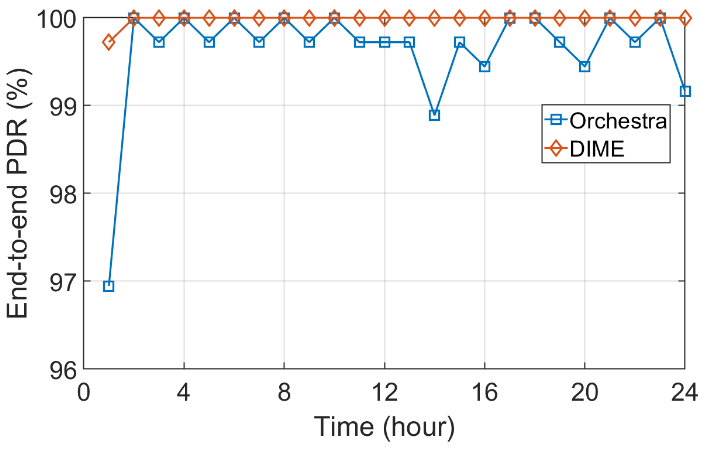
Figure 6.6: Performance comparison under the low data rate setting.

Figure 6.6(c) plots the CDF of the radio duty cycle of all field nodes under DIME and Orchestra, where the two approaches deliver similar performance. The median duty cycle is 4.44% under DIME and 4.48% under Orchestra. These results clearly demonstrate the benefits of DIME on improving the reliability and reducing the latency without the overhead of increased radio duty cycle.

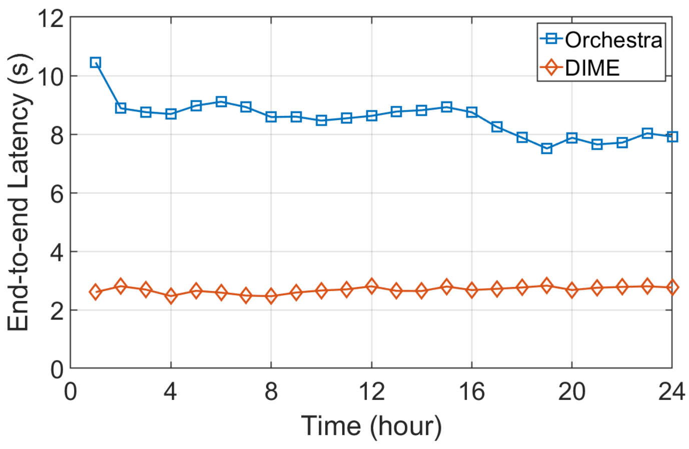
Figure 6.7 compares the performance of the entire network under DIME and Orchestra over 24 hours. As Figure 6.7(a) shows, DIME achieves the PDR of 99.7% at the first hour and always delivers 100% PDR after the first hour, while Orchestra experiences the lowest PDR of 96.9% at the first hour and achieves the PDR above 98.8% after that. As Figure 6.7(b) shows, DIME always keeps the latency less than 2.82s, while Orchestra experiences the latency up to 10.45s. These results show that DIME constantly outperforms Orchestra over a long time of operation at the low data rate.

6.4.4 Performance at Network Initialization

To observe the network performance at startup, we measure end-to-end PDR and end-to-end latency of all application packets within the first 600s after all devices initially start and operate under the high data rate setting using DIME and Orchestra, respectively. Figure 6.8 shows the network performance over time, where each data point indicates the performance in a 30s duration. As shown in Figure 6.8(a) and Figure 6.8(b), DIME's performance stabilizes quickly with the PDR always above 91.7% and the latency less than 4.84s. On the other hand, Orchestra experiences the lowest PDR of 54.2% and the latency up to 37.87s at the beginning of operation. Compared to Orchestra, DIME saves at least 300s at startup to operate reliably and stably. DIME's better performance at startup is

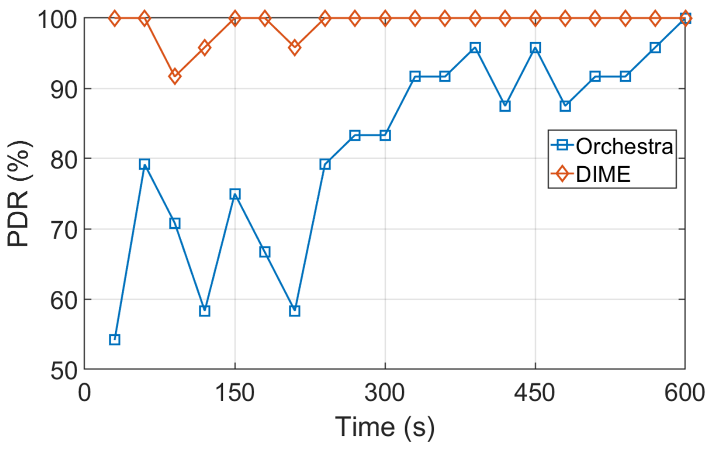


(a) End-to-end PDR.

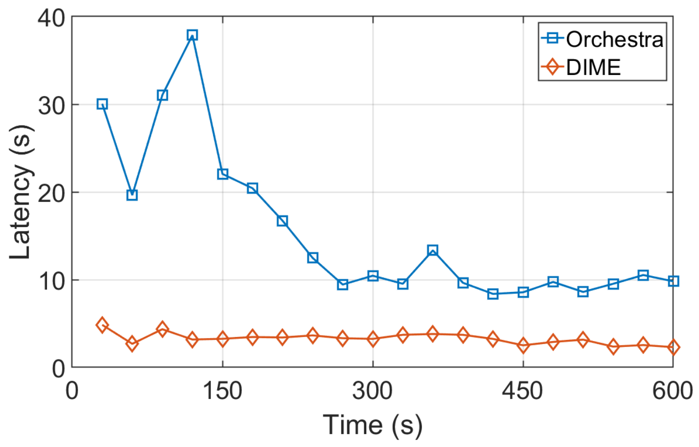


(b) End-to-end latency.

Figure 6.7: Network performance over 24 hours under the low data rate setting.



(a) End-to-end PDR.



(b) End-to-end latency.

Figure 6.8: Network performance at startup.

benefit from the direct synchronization of all nodes in the network, which reduces the network management complexity.

6.5 Conclusion

Industrial wireless networks often suffer high network management complexity due to a large number of network devices and a mesh network topology. To mitigate the management complexity of industrial wireless networks, we develop the DIME system that leverages the CTC technique to efficiently synchronize the network and enable direct message delivery through long distance links instead of hop-by-hop transports. Testbed experiments show that DIME significantly improves the end-to-end PDR, reduces the end-to-end latency, and saves time at startup to stabilize the network performance compared to Orchestra.

7 Conclusion

Various wireless technologies are readily available to support communication between devices in IoT applications. However, it is very challenging to achieve reliable and efficient data transfer through wireless medium due to its inherent unreliability. The data transfer challenge is significantly amplified by a number of factors that exist in IoT applications, including the uncertainties of network traffic and wireless environment, the real-time requirement of data delivery, the mobility of IoT platforms, and the management complexity of industrial wireless networks. Fortunately, heterogeneous radios and new wireless technologies are becoming increasingly available on modern IoT devices, offering new opportunities to address the data transfer challenge.

To adapt to the uncertainties of network traffic and wireless environment, we develop the ARTPoS system that leverages multiple heterogeneous radios and selects the most suitable radio(s) and transmission power(s) for the current conditions and requirements. Experimental results show that ARTPoS can save more than $100mW$ of power consumption over fixed-power and single-radio solutions under various network traffic and wireless environment, while maintaining satisfactory link reliability. To support real-time data deliveries energy-efficiently, we develop the RRaSB system that runs on our embedded platform equipped with multiple heterogeneous radios and allows dynamic radio switching and bundling among them based on our real-time radio selection and data partitioning algorithms. Experimental results show that RRaSB significantly outperforms GreenBag by reducing more than $300mJ$ of energy consumption in the real-time transfer of 109KB data under various deadlines, while meeting satisfactory real-time requirements. To support mobile IoT platforms, we develop the ShuttleNet system that collects real-time data from running vehicles based on the LPWAN technology and controls the network's physical-layer parameters at runtime. Real-world experiments show that ShuttleNet increases the data collection throughput by 58.6% and improves the network reliability by 27.0% compared to the existing methods. To mitigate the management complexity of industrial wireless networks, we develop the DIME system that leverages the CTC technique to efficiently synchronize the network and enable direct message delivery through long distance links instead of hop-by-hop transports. Testbed experiments show that DIME significantly improves the end-to-end PDR from 92.6% to 99.8%, reduces the end-to-end latency from $9.03s$ to $3.06s$, and saves $300s$ at startup to stabilize the network performance at the high data rate compared to the state of the art.

Bibliography

[1] D. Mu et al. “Adaptive Radio and Transmission Power Selection for Internet of Things”. In: *ACM/IEEE International Symposium on Quality of Service (IWQoS)*. 2017.

[2] Di Mu et al. “Robust Optimal Selection of Radio Type and Transmission Power for Internet of Things”. In: *ACM Transactions on Sensor Networks (TOSN)* 15.4 (2019).

[3] Di Mu et al. “Energy-Efficient Radio Selection and Data Partitioning for Real-Time Data Transfer”. In: *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2019.

[4] Di Mu et al. “Radio Selection and Data Partitioning for Energy-Efficient Wireless Data Transfer in Real-Time IoT Applications”. In: *Ad Hoc Networks* 107 (2020).

[5] Di Mu et al. “Runtime Control of LoRa Spreading Factor for Campus Shuttle Monitoring”. In: *IEEE International Conference on Network Protocols (ICNP)*. 2020.

[6] Karim Habak, Khaled A Harras, and Moustafa Youssef. “Bandwidth Aggregation Techniques in Heterogeneous Multi-homed Devices: A Survey”. In: *Computer Networks* 92.1 (2015), pp. 168–188.

[7] Internet Engineering Task Force. *RFC 6824 - Multipath TCP*. 2013. URL: `https://tools.ietf.org/html/rfc6824`.

[8] Ashkan Nikravesh et al. "An In-depth Understanding of Multipath TCP on Mobile Devices: Measurement and System Design". In: *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. MobiCom '16. New York City, NY, USA: ACM, 2016, pp. 189–201.

[9] Mohammad Javad Shamani, Weiping Zhu, and Saeid Rezaie. "On the Energy Inefficiency of MPTCP for Mobile Computing". In: *International Conference on Wired/Wireless Internet Communication (WWIC)*. 2016.

[10] Cheng-Lin Tsao and Raghupathy Sivakumar. "On Effectively Exploiting Multiple Wireless Interfaces in Mobile Hosts". In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '09. Rome, Italy: ACM, 2009, pp. 337–348.

[11] Duc Hoang Bui et al. "GreenBag: Energy-efficient Bandwidth Aggregation For Real-time Streaming in Heterogeneous Mobile Wireless Networks". In: *Proceedings of the 34th IEEE Real-Time Systems Symposium*. RTSS '13. Vancouver, BC, Canada: IEEE, 2013, pp. 57–67.

[12] Shan Lin et al. "ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks". In: *ACM Transactions on Sensor Networks* 12.1 (2016), pp. 6–19.

[13] Shan Lin et al. "Towards Stable Network Performance in Wireless Sensor Networks". In: *Proceedings of the 30th IEEE Real-Time Systems Symposium*. RTSS '09. Washington, DC, USA: IEEE, 2009, pp. 227–237.

[14] Gregory Hackmann, Octav Chipara, and Chenyang Lu. “Robust Topology Control for Indoor Wireless Sensor Networks”. In: *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*. SenSys ’08. Raleigh, NC, USA: ACM, 2008, pp. 57–70.

[15] Yong Fu et al. “Practical Control of Transmission Power for Wireless Sensor Networks”. In: *Proceedings of the 20th IEEE International Conference on Network Protocols*. ICNP ’12. Austin, TX, USA: IEEE, 2012, pp. 1–10.

[16] Martin Burkhart et al. “Does Topology Control Reduce Interference?” In: *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. MobiHoc ’04. Roppongi Hills, Tokyo, Japan: ACM, 2004, pp. 9–19.

[17] Yan Gao, Jennifer C Hou, and Hoang Nguyen. “Topology Control for Maintaining Network Connectivity and Maximizing Network Capacity under the Physical Model”. In: *Proceedings of the 27th Conference on Computer Communications*. INFOCOM ’08. Phoenix, AZ, USA: IEEE, 2008, pp. 1013–1021.

[18] Yeon-sup Lim et al. “Design, Implementation, and Evaluation of Energy-Aware Multi-Path TCP”. In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. CoNEXT ’15. Heidelberg, Germany: ACM, 2015, p. 30.

[19] Ana Nika et al. “Energy and Performance of Smartphone Radio Bundling in Outdoor Environments”. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15. Florence, Italy: ACM, 2015, pp. 809–819.

[20] Jiyan Wu et al. “Energy-Efficient Bandwidth Aggregation for Delay-Constrained Video over Heterogeneous Wireless Networks”. In: *IEEE Journal on Selected Areas in Communications* 35.1 (2017), pp. 30–49.

[21] Chaojie Gu et al. “One-Hop Out-of-Band Control Planes for Low-Power Multi-Hop Wireless Networks”. In: *IEEE Conference on Computer Communications (INFOCOM)*. 2018.

[22] Romain Jacob et al. “End-to-End Real-Time Guarantees in Wireless Cyber-Physical Systems”. In: *IEEE Real-Time Systems Symposium (RTSS)*. 2016.

[23] Tianyu Zhang et al. “FD-PaS: A Fully Distributed Packet Scheduling Framework for Handling Disturbances in Real-Time Wireless Networks”. In: *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2018.

[24] Chengjie Wu et al. “Real-Time Wireless Routing for Industrial Internet of Things”. In: *IEEE/ACM International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2018.

[25] Giuliana Iapichino et al. “Advanced Hybrid Satellite and Terrestrial System Architecture for Emergency Mobile Communications”. In: *International Communications Satellite Systems Conference (ICSSC)*. 2008.

[26] Tao Tang et al. “Field Test Results Analysis in Urban Rail Transit Train Ground Communication Systems of Integrated Service Using LTE-M”. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2016.

[27] Arwa Khayat et al. “LTE Based Telecommunication System for Urban-Guided Transports”. In: *Transport Research Arena (TRA)*. 2014.

[28] Shanzhi Chen et al. “Vehicle-to-Everything (V2X) Services Supported by LTE-Based Systems and 5G”. In: *IEEE Communications Standards Magazine* 1.2 (2017).

[29] Md Tamzeed Islam, Bashima Islam, and Shahriar Nirjon. “Duty-Cycle-Aware Real-Time Scheduling of Wireless Links in Low Power WANs”. In: *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2018.

[30] Jansen C Liando et al. “Known and Unknown Facts of LoRa: Experiences from a Large-Scale Measurement Study”. In: *ACM Transactions on Sensor Networks* 15.2 (2019).

[31] José Santa et al. “LPWAN-Based Vehicular Monitoring Platform with a Generic IP Network Interface”. In: *Sensors* 19.2 (2019).

[32] Ricardo Salazar-Cabrera, Álvaro Pachón de la Cruz, and Juan Manuel Madrid Molina. “Proof of Concept of an IoT-Based Public Vehicle Tracking System, Using LoRa (Long Range) and Intelligent Transportation System (ITS) Services”. In: *Journal of Computer Networks and Communications* (2019).

[33] Takuya Boshita, Hidekazu Suzuki, and Yukimasa Matsumoto. “IoT-Based Bus Location System Using LoRaWAN”. In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[34] Pengxin Guan et al. “A Real-Time Bus Positioning System Based on LoRa Technology”. In: *IEEE International Conference on Smart Grid and Smart Cities (ICSGSC)*. 2018.

[35] Silvano Bertoldo et al. “Feasibility Analysis of a LoRa-Based WSN Using Public Transport”. In: *Applied System Innovation* 1.4 (2018).

[36] Ange Ouya et al. “An Efficient Electric Vehicle Charging Architecture Based on LoRa Communication”. In: *IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2017.

[37] *LoRaWAN Adaptive Data Rate*. URL: `https://www.thethingsnetwork.org/`.

[38] Mariusz Slabicki, Gopika Premsankar, and Mario Di Francesco. “Adaptive Configuration of LoRa Networks for Dense IoT Deployments”. In: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 2018.

[39] Martin Bor and Utz Roedig. “LoRa Transmission Parameter Selection”. In: *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2017.

[40] Muhammad Asad Ullah et al. “K-Means Spreading Factor Allocation for Large-Scale LoRa Networks”. In: *Sensors* 19.21 (2019).

[41] Silvia Demetri et al. “Automated Estimation of Link Quality for LoRa: a Remote Sensing Approach”. In: *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2019.

[42] Norhane Benkahla et al. “Enhanced ADR for LoRaWAN Networks with Mobility”. In: *IEEE International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2019.

[43] Padraig Cunningham and Sarah Jane Delany. “k-Nearest Neighbour Classifiers”. In: *Multiple Classifier Systems* 34.8 (2007).

[44] Feng Yu et al. “5G WiFi Signal-Based Indoor Localization System Using Cluster k-Nearest Neighbor Algorithm”. In: *International Journal of Distributed Sensor Networks* 10.12 (2014).

[45] Azin Arya, Philippe Godlewski, and Philippe Mellé. “Performance Analysis of Outdoor Localization Systems Based on RSS Fingerprinting”. In: *International Symposium on Wireless Communication Systems (ISWCS)*. 2009.

[46] Wenchao Li et al. “A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network”. In: *Journal of Electrical and Computer Engineering* (2014).

[47] Liqiang Pan and Jianzhong Li. “K-Nearest Neighbor Based Missing Data Estimation Algorithm in Wireless Sensor Networks”. In: *Wireless Sensor Network* 2.02 (2010).

[48] Brad K Donohoo et al. “Context-Aware Energy Enhancements for Smart Mobile Devices”. In: *IEEE Transactions on Mobile Computing* 13.8 (2013).

[49] Yunqian Ma. “Improving Wireless Link Delivery Ratio Classification with Packet SNR”. In: *IEEE International Conference on Electro/Information Technology (EIT)*. 2005.

[50] *IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH)*. URL: https://datatracker.ietf.org/doc/html/rfc7554.

[51] Simon Duquennoy et al. "Orchestra: Robust Mesh Networks through Autonomously Scheduled TSCH". In: *ACM Conference on Embedded Networked Sensor Systems*. 2015.

[52] Seohyang Kim, Hyung-Sin Kim, and Chongkwon Kim. "ALICE: Autonomous Link-based Cell Scheduling for TSCH". In: *International Conference on Information Processing in Sensor Networks*. 2019.

[53] Junyang Shi, Mo Sha, and Zhicheng Yang. "DiGS: Distributed Graph Routing and Scheduling for Industrial Wireless Sensor-Actuator Networks". In: *IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2018.

[54] Junyang Shi, Di Mu, and Mo Sha. "LoRaBee: Cross-Technology Communication from LoRa to ZigBee via Payload Encoding". In: *IEEE International Conference on Network Protocols (ICNP)*. 2019.

[55] Junyang Shi, Xingjian Chen, and Mo Sha. "Enabling Direct Messaging from LoRa to ZigBee in the 2.4 GHz Band for Industrial Wireless Networks". In: *IEEE International Conference on Industrial Internet (ICII)*. 2019.

[56] Mo Sha et al. "Self-Adapting MAC Layer for Wireless Sensor Networks". In: *Proceedings of the 34th IEEE Real-Time Systems Symposium*. RTSS '13. Vancouver, BC, Canada: IEEE, 2013, pp. 192–201.

[57] Michael P Andersen, Gabe Fierro, and David E Culler. "System Design for a Synergistic, Low Power Mote/BLE Embedded Platform". In: *Proceedings*

of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks. IPSN '16. Vienna, Austria: IEEE, 2016, pp. 1–12.

[58] TI.com. *CC2650 SimpleLink Multi-Standard 2.4 GHz Ultra-Low Power Wireless MCU.* 2016. URL: `http://www.ti.com/product/CC2650`.

[59] Raspberry Pi. *Raspberry Pi 3 Model B.* 2016. URL: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[60] CompuLab Ltd. *IOT-GATE-iMX7 - Industrial Internet of Things Gateway.* 2018. URL: `https://www.compulab.com/products/iot-gateways/iot-gate-imx7-nxp-i-mx-7-internet-of-things-gateway/`.

[61] Ruogu Zhou et al. "ZiFi: Wireless LAN Discovery via ZigBee Interference Signatures". In: *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking.* MobiCom '10. Chicago, Illinois, USA: ACM, 2010, pp. 49–60.

[62] Monsoon Solutions. *LVPM Product Documentation.* 2014. URL: `https://www.msoon.com/lvpm-product-documentation`.

[63] Raspbian. *Welcome to Raspbian.* 2012. URL: `https://www.raspbian.org/`.

[64] Contiki. *Contiki Operating System.* 2018. URL: `https://github.com/contiki-os/contiki/`.

[65] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245. ISSN: 00401706. URL: `http://www.jstor.org/stable/1268522`.

[66] Jeong-Soo Park. “Optimal Latin-hypercube Designs for Computer Experiments”. In: *Journal of Statistical Planning and Inference* 39.1 (1994), pp. 95–111.

[67] R Tyrrell Rockafellar and Johannes O Royset. “Engineering Decisions under Risk Averseness”. In: *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering* 1.2 (2015), p. 04015003.

[68] Jianmin Jia and James S Dyer. “A standard measure of risk and risk-value models”. In: *Management Science* 42.12 (1996), pp. 1691–1705.

[69] R Tyrrell Rockafellar and Stanislav Uryasev. “Conditional Value-at-risk for General Loss Distributions”. In: *Journal of Banking & Finance* 26.7 (2002), pp. 1443–1471.

[70] Giuseppe Cardillo. *Four Parameters Logistic Regression - There and Back Again*. online. Apr. 2013.

[71] Michael Junger et al., eds. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Heidelberg, Germany: Springer, 2009. Chap. Nonlinear Integer Programming. ISBN: 978-3-540-68279-0.

[72] Carlo Alberto Boano et al. “Jamlab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation”. In: *International Conference on Information Processing in Sensor Networks*. IPSN ’11. Chicago, IL, USA: ACM, 2011.

[73] MEMSIC. *TELOSB MOTE PLATFORM*. 2014. URL: `http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf`.

[74] J. Ruminski et al. “Interactions with Recognized Patients Using Smart Glasses”. In: *IEEE International Conference on Human System Interactions (HSI).* 2015.

[75] C. Eling, L. Klingbeil, and H. Kuhlmann. “A Direct Georeferencing System for Real-Time Position and Attitude Determination of Lightweight UAVs”. In: *FIG Working Week.* 2015.

[76] J. Lynch, C. Rarrar, and J. Michaels. “Structural Health Monitoring: Technological Advances to Practical Implementations”. In: *Proceedings of the IEEE, Special Issue on Structural Health Monitoring* 104.8 (2016), pp. 1508–1512.

[77] Hoa Hong Nguyen et al. “A Review on IoT Healthcare Monitoring Applications and a Vision for Transforming Sensor Data into Real-Time Clinical Feedback”. In: *IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD).* 2017.

[78] Chenyang Lu et al. “Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems”. In: *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems* 104.5 (2015), pp. 1013–1024.

[79] T. Watteyne et al. “Industrial Wireless IP-Based Cyber Physical Systems”. In: *Proceedings of the IEEE, Special Issue on Industrial Cyber Physical Systems* 104.5 (2016), pp. 1025–1038.

[80] *LX IoT Cores.* URL: `https://lx-group.com.au/iot-cores/`.

[81] Prajakta S Kalekar. “Time Series Forecasting Using Holt-Winters Exponential Smoothing”. In: *Kanwal Rekhi School of Information Technology* (2004).

[82] *GNU Linear Programming Kit (GLPK).* URL: `https://www.gnu.org/software/glpk/`.

[83] *Analyzing Performance for Amazon Rekognition Apps.* URL: `https://aws.amazon.com/blogs/compute/analyzing-performance-for-amazon-rekognition-apps-written-on-aws-lambda-using-aws-x-ray/`.

[84] Beatriz Soret et al. "Fundamental Tradeoffs among Reliability, Latency and Throughput in Cellular Networks". In: *IEEE Globecom Workshops.* 2014.

[85] *LoRa Modulation Basics.* URL: `https://www.semtech.com/`.

[86] Ferran Adelantado et al. "Understanding the Limits of LoRaWAN". In: *IEEE Communications Magazine* 55.9 (2017).

[87] François Delobel, Nancy El Rachkidy, and Alexandre Guitton. "Analysis of the Delay of Confirmed Downlink Frames in Class B of LoRaWAN". In: *IEEE Vehicular Technology Conference (VTC Spring).* 2017.

[88] Affan A Syed et al. "Understanding Spatio-Temporal Uncertainty in Medium Access with ALOHA Protocols". In: *Proceedings of Workshop on Underwater Networks (WuWNet).* 2007.

[89] *RN2903 Provided by MICROCHIP.* URL: `https://www.microchip.com/wwwproducts/en/RN2903`.

[90] *iC880A Provided by IMST.* URL: `https://www.wireless-solutions.de/products/radiomodules/ic880a.html`.

[91] Orion Afisiadis et al. "On the Error Rate of the LoRa Modulation with Interference". In: *Transactions on Wireless Communications* (2019).

[92] Semtech. *SX1272 Datasheet*. URL: https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272.

[93] Junyang Shi and Mo Sha. “Parameter Self-Configuration and Self-Adaptation in Industrial Wireless Sensor-Actuator Networks”. In: *IEEE Conference on Computer Communications (INFOCOM)*. 2019.

[94] *LoRa*. URL: https://lora-alliance.org.