공학박사 학위논문

# BLE Connectivity and its Multi-hop Extension for IoT Applications

다양한 사물 인터넷 서비스를 위한
**Bluetooth Low Energy** 기술의
연결성 및 멀티 홉 확장

2017년 8월

서울대학교 대학원
전기.컴퓨터 공학부
이 태 섭

# BLE Connectivity and its Multi-hop Extension for IoT Applications

지도교수 박 세 웅

이 논문을 공학박사 학위논문으로 제출함
2017년 6월

서울대학교 대학원
전기.컴퓨터 공학부
이 태 섭

이태섭의 박사 학위논문을 인준함
2017년 6월

| | | |
|---|---|---|
| 위 원 장 | 김 종 권 | (인) [illegible] |
| 부위원장 | 박 세 웅 | (인) [illegible] |
| 위 원 | 권 태 경 | (인) [illegible] |
| 위 원 | 최 성 현 | (인) [illegible] |
| 위 원 | 윤 성 국 | (인) [illegible] |

# Abstract

Bluetooth Low Energy (BLE) is one of the representative low-power communication protocols that are being used to provide wireless connectivity for resource constrained devices as part of Internet of Things (IoT). Despite its commercial adoption, BLE's current use is limited to short-range applications due to the lack of research about its coverage extension. In this dissertation, we investigate two issues that need to be addressed for BLE's network coverage extension and also consider a new application scenario using a BLE-based multi-hop network.

First, we tackle the BLE connection maintenance and energy consumption problems by adaptively controlling one of BLE's link layer parameters ($T_{CI}$) under dynamic channel condition. We formulate an optimization problem to find an optimal $T_{CI}$ and design a connection interval adaptation mechanism for BLE to achieve high energy efficiency while maintaining robust connectivity. We evaluate our proposed solutions through testbed experiments and simulation which shows that it reduces energy consumption of BLE in dynamic channel environments.

Secondly, we consider a protocol architecture that aims to run IPv6 routing protocol for low power and lossy networks (RPL) over BLE to construct BLE-based multi-hop networks. We design an adaptation layer between BLE and RPL which tightly couples RPL and BLE operation. We implement the adaptation layer in a Linux kernel to realize RPL over BLE. Through extensive experiments in an indoor testbed, we evaluate the performance of RPL over BLE and compare the performance results with that of RPL over IEEE 802.15.4 which shows significant improvement.

Lastly, we consider a new application scenario of BLE using the coverage extension of BLE based on multi-hop networking. We propose a novel layered architecture of Wi-Fi and BLE that constructs an energy efficient and high data rate supportable ad-hoc network for disaster communication. We implement the proposed architecture in Linux kernel and evaluate the performance through our indoor testbed. The result shows that our proposed solution reduces the average power consumption of nodes in the testbed compared to a conventional Wi-Fi ad-hoc network.

**Keywords:** Bluetooth Low Energy (BLE), Internet of Things (IoT), connection interval, multi-hop network, disaster communication

**Student Number:** 2011-20913

# Contents

**1 Introduction** **1**

1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . 4

1.2.1 Low power consumption of BLE . . . . . . . . 4

1.2.2 BLE multi-hop networking . . . . . . . . . . . 5

1.3 Contributions and Outline . . . . . . . . . . . . . . . . . 6

**2 CABLE: Connection Interval Adaptation for BLE in Dynamic Wireless Environments** **10**

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Background and Problem Statement . . . . . . . . . . 14

2.2.1 Link layer operation . . . . . . . . . . . . . . . 14

2.2.2 Connection loss due to supervision timeout . . 16

2.2.3 BLE protocol stack and connection interval setting . . . . . . . . . . . . . . . . . . . . . . . . 17

2.2.4 Problem of BLE with fixed connection interval 19

2.3 Connection Interval Optimization . . . . . . . . . . . . 22

2.3.1 Problem formulation . . . . . . . . . . . . . . . . 22
2.3.2 Problem solution . . . . . . . . . . . . . . . . . . . 26
2.4 CABLE System Design . . . . . . . . . . . . . . . . . 29
2.4.1 PER estimator . . . . . . . . . . . . . . . . . . . . 30
2.4.2 $T_{CI}$ adjuster . . . . . . . . . . . . . . . . . . . . 33
2.5 Performance Evaluation . . . . . . . . . . . . . . . . . 34
2.5.1 Simulation results . . . . . . . . . . . . . . . . . . 35
2.5.2 Experimental results . . . . . . . . . . . . . . . . 40
2.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . 41

**3 A Synergistic Architecture for RPL over BLE 43**

3.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . 43
3.2 Background . . . . . . . . . . . . . . . . . . . . . . . 47
3.2.1 RPL operation . . . . . . . . . . . . . . . . . . . . 47
3.2.2 BLE link layer operation . . . . . . . . . . . . . . 48
3.2.3 6LoWPAN for BLE . . . . . . . . . . . . . . . . . 50
3.3 Design of RPL over BLE . . . . . . . . . . . . . . . . 52
3.3.1 Synergistic Network Architecture for RPL over BLE . . . . . . . . . . . . . . . . . . . . . . . . 52
3.3.2 DIO broadcast over advertising channels . . . . 54
3.3.3 Routing metric for RPL over BLE . . . . . . . 57
3.3.4 RPL parent change with BLE connection management . . . . . . . . . . . . . . . . . . . . . . 60
3.4 ALBER Implementation . . . . . . . . . . . . . . . . 61

3.5 Performance Evaluation . . . . . . . . . . . . . . . . . . 64
3.5.1 Testbed environments . . . . . . . . . . . . . . 64
3.5.2 Comparison of RPL over BLE vs. RPL over 802.15.4 . . . . . . . . . . . . . . . . . . . . . . . . 65
3.5.3 Effect of varying connection interval . . . . . . 70
3.5.4 Effect of ECI-based routing metric . . . . . . . 71
3.6 Summary . . . . . . . . . . . . . . . . . . . . . . . . . 73

**4 Wi-BLE: A Novel Layered Architecture of Wi-Fi & BLE Networks for Disaster Communications 74**
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . 74
4.2 Background . . . . . . . . . . . . . . . . . . . . . . . . 78
4.2.1 Application requirements of ad-hoc networks for disaster communications . . . . . . . . . . . . . 78
4.2.2 Candidate wireless interfaces for ad-hoc networks 79
4.2.3 Wi-BLE use scenario . . . . . . . . . . . . . . . . 79
4.3 Wi-BLE System Overview . . . . . . . . . . . . . . . . 80
4.3.1 Protocol Architecture . . . . . . . . . . . . . . . 80
4.3.2 Operation Overview . . . . . . . . . . . . . . . . 81
4.4 MABLE: Mobile Ad-hoc for BLE . . . . . . . . . . . . 82
4.4.1 Routing protocol selection for MABLE . . . . . 82
4.4.2 BLE Channel Usage for AODV over BLE . . . 84
4.5 Wi-BLE: Wi-Fi Ad-hoc over BLE networks . . . . . . 88
4.5.1 Wi-BLE control packet delivery over BLE path 88

4.5.2 Routing protocol for Wi-BLE . . . . . . . . . . 89

4.5.3 Wi-Fi on/off control for energy saving . . . . . 92

4.6 Implementation . . . . . . . . . . . . . . . . . . . . . . . 92

4.7 Performance Evaluation . . . . . . . . . . . . . . . . . 94

4.7.1 Testbed Environments . . . . . . . . . . . . . . 94

4.7.2 Hop distance & Throughput . . . . . . . . . . . 95

4.7.3 Power Consumption . . . . . . . . . . . . . . . 97

4.8 Summary . . . . . . . . . . . . . . . . . . . . . . . . . 98

**5 Conclusion** **100**

5.1 Research Contributions . . . . . . . . . . . . . . . . . 100

5.2 Further Research Direction . . . . . . . . . . . . . . . 102

# List of Tables

2.1 Parameters for BLE operation . . . . . . . . . . . . . . . 35

3.1 HCI commands for each operation of *RPL over BLE* 62

4.1 Comparison between Wi-Fi and BLE . . . . . . . . . . 75

# List of Figures

1.1 The portion of BLE-based IoT products in smart home market [1]. . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.1 BLE link layer operations. . . . . . . . . . . . . . . . . . . 15

2.2 BLE protocol stack and determination of Connection Interval. . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.3 Measurement topology. . . . . . . . . . . . . . . . . . . . . 19

2.4 PER and the number of connection losses at 30 minute intervals for links 0 (best case) and 2 (worst case). . . 20

2.5 State diagram to compute the average interarrival time of supervision timeout (i.e., $ST_{avg}(T_{CI})$). . . . . . . . . 23

2.6 Power consumption of BLE for maintaining connectivity as a function of $T_{CI}$ (i.e., $P(T_{CI})$). . . . . . . . . . 25

2.7 Overview of the *CABLE* system. . . . . . . . . . . . 29

2.8 CDF of Logical Link layer ping's RTT with $T_{CI}$ = 1sec. 31

2.9 PER estimation based on L2CAP ping's RTT. . . . . 31

2.10 Simulation results for two different channel scenarios; figure (a) for the same and static channel PER scenario, and figures (b)$_\sim$(d) for a real channel trace scenario. . 37
2.11 Experimental results from the indoor office testbed. . 40

3.1 Link layer operation of BLE in each channel type. . . 49
3.2 Design overview of the proposed architecture for RPL over BLE. . . . . . . . . . . . . . . . . . . . . . . . . . 51
3.3 Exchange of RPL control messages over advertising channels with low power listening. . . . . . . . . . . 54
3.4 CDF of Logical Link layer ping's RTT with $T_{CI} = 1sec$. 57
3.5 Protocol stack of *RPL over BLE* including *ALBER*. . 60
3.6 Connection-aware parent change procedures in *RPL over BLE*. . . . . . . . . . . . . . . . . . . . . . . . . . 61
3.7 Testbed topology map with 29 nodes and one root. . . 63
3.8 TelosB for IEEE 802.15.4 node (left), a set of Raspberry Pi and BCM4356 for BLE node (middle), and NI USB-6210 for power meter (right). . . . . . . . . . . . . . . 64
3.9 Performance metrics of *RPL over BLE* and *RPL over IEEE 802.15.4* for 24 hours in a light traffic scenario (5 minutes/packet/node). . . . . . . . . . . . . . . . . 66
3.10 Performance comparison of *RPL over BLE* vs. *RPL over IEEE 802.15.4* with varying traffic load. . . . . 68

3.11 Performance of *RPL over BLE* with varying connection interval in a heavy traffic scenario (30 seconds/packet/node). 70

3.12 Performance of *RPL over BLE* with and without our ECI-based routing metric, for 24 hours in a light traffic scenario (5 minutes/packet/node). . . . . . . . . . . . . 72

4.1 Protocol stack of *Wi-BLE*. . . . . . . . . . . . . . . . . 80

4.2 Operation overview of *Wi-BLE*. . . . . . . . . . . . . . . 81

4.3 End-to-end connection establishment of BLE nodes. . 86

4.4 Wi-BLE control packet format. . . . . . . . . . . . . . . 88

4.5 Topology comparison of USE_BLE_ROUTE vs. USE_WLAN_AODV. 90

4.6 Testbed topology map with 10 nodes. . . . . . . . . . 94

4.7 Performance of Wi-Fi ad-hoc network with three different schemes. . . . . . . . . . . . . . . . . . . . . . . . 95

4.8 The average power consumption of nodes vs. flow arrival rate. . . . . . . . . . . . . . . . . . . . . . . . . . 97

# Chapter 1

# Introduction

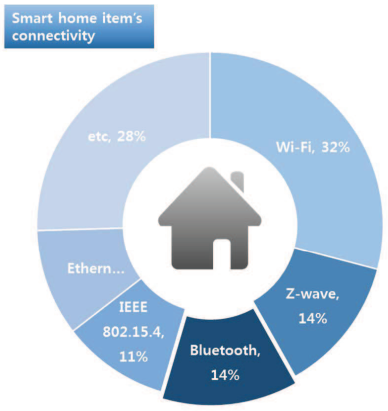
## 1.1 Motivation

Since BLE was first introduced with the adoption of the Bluetooth core 4.0 specification, it has shown steady growth in IoT markets due to its advantages over other low-power wireless protocols such as IEEE 802.15.4 and Z-Wave. The pertinent features are:

- Strong PHY/MAC performance: PHY and MAC layers of BLE provide higher data rate and better reliability than competing low-power wireless technologies. Specifically, BLE's PHY layer supports data rates up to 1Mbps that are significantly higher than that of IEEE 802.15.4 (250kbps) and Z-Wave (100kbps). BLE's MAC layer exploits an adaptive frequency hopping (AFH) mechanism which enables it to resiliently coexist with other wireless devices in the 2.4 GHz ISM (unlicensed) band. BLE achieves these improvements with lower energy consumption which makes it well-suited for IoT.

- High accessibility: In contrast to other IoT wireless protocols, BLE is already embedded in today's ubiquitous devices such as smartphones and tablets. Employing these devices as gateways obviates the need for dedicated gateways for Internet connectivity. This relieves the burden of installation and management of IoT gateways and reduces cost. High accessibility also facilitates smartphone/tablet application development over BLE.
- High familiarity: To develop technologies *actually used* in our daily lives, it is important to consider practical matters such as public awareness and industry trends which are sometimes more important than technical performance. Even before the introduction of BLE, Bluetooth was an established technology widely used in applications such as wireless headset, mouse and keyboard. Familiarity gives BLE an advantage over IEEE 802.15.4 and Z-Wave yet.

Despite the advantages, for new promising application domains such as smart homes and buildings that need to cover a larger area than conventional short-range applications, the current BLE is insufficient. Specifically, BLE's practical transmission range is not enough to provide home area connectivity. Figure 1.1 shows the market share of wireless technologies for smart home applications in 2016. Despite



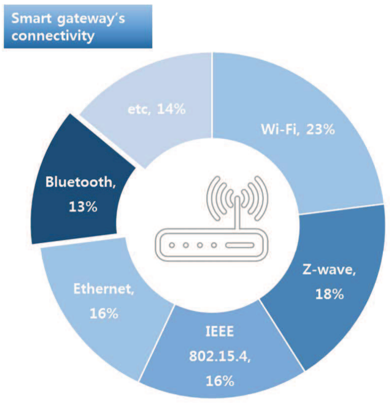


Figure 1.1: The portion of BLE-based IoT products in smart home market [1].

BLE's large deployment base, with respect to smart home applications, we observe that BLE is on par with or lower than 802.15.4 and Z-wave. Considering the large gap between the annual chipset shipment of BLE (3 billion) and IEEE 802.15.4 (0.85 billion), the market share results indicate that BLE chipsets are mostly being used for short-range applications.

This thesis is motivated by the question: *"Is it possible for BLE to show its significant potential for supporting longer range application?"* If so, its high accessibility and familiarity provide an advantage over other low power communication protocols in the various IoT application domains. This dissertation will answer this question considering two issues: link layer optimization under dynamic channel condition and multi-hop networking and also suggest a new application scenario based on the extended coverage of BLE network.

## 1.2 Related Work

### 1.2.1 Low power consumption of BLE

One of the most important characteristics of BLE is low energy consumption. On the experimental side, Kamath and Lindh measured energy consumption of BLE for each specific operation by using a BLE radio chip (CC2541) and calculated battery lifetime for several application scenarios [2]. Siekkinen *et al.* designed an energy consumption model for BLE based on measurement results [3]. They performed comparisons with IEEE 802.15.4 and showed that BLE provides lower energy consumption and better energy efficiency (J/bit). In [4] and [5], the authors implemented a BLE-based multi-hop network using a routing protocol for low power lossy networks (RPL). They compared it with RPL over IEEE 802.15.4 and showed improved packet delivery performance with respect to energy efficiency and reliability. On the theoretical front, Liu *et al.* analyzed energy and latency performance of BLE during device discovery procedures [6][7]. They showed that connection establishment/recovery dominates other operations with respect to energy consumption. A precise energy model of BLE was advanced by Kindt *et al.* in [8]. These works help reveal BLE's characteristics in terms of energy consumption.

Recently, Kindt *et al.* proposed a power management scheme for BLE that dynamically adjusts $T_{CI}$ based on application data rate under ideal channel conditions [9]. When throughput exceeds data rate, energy may be saved by increasing $T_{CI}$ and vice versa. In [10], the authors presented an analytical model for achieving maximum BLE throughput in error-prone link environments. They provided analytic results on the relation between $T_{CI}$ and throughput but did not investigate the impact of $T_{CI}$ on connection maintenance and energy consumption in dynamic wireless environments. In this dissertation, we focus on the important role connection maintenance plays on energy consumption, and propose adaptive connection interval control under dynamic channel conditions.

### 1.2.2 BLE multi-hop networking

Some recent work looked at constructing a multi-hop BLE network which may be classified into two groups: flooding- vs. connection-based approaches. As an example of the first approach, Cambridge Silicon Radio (CSR) developed a flooding-based BLE mesh network, termed *CSRmesh* [11]. Nordic Semiconductor developed a BLE-based multi-hop mesh network by using a flood control algorithm called *Trickle* [12][13]. Bluetooth SIG organized a smart mesh working group for devising a new specification of a flooding-based BLE mesh network [14]. These efforts represent growing industry interest to invest in BLE-based multi-hop networks and provide an initial evidence of its feasibility. However, flooding-based approaches cannot exploit advantages of BLE's connection-based operation which include channel hopping, synchronous data exchange, and link level retransmission.

In the second group, several studies explored feasibility of a connection-based multi-hop BLE network [15][16][17]. A limitation of these works is that they tested basic operations without providing comprehensive performance evaluation. They also did not consider RPL over BLE issues and their impact of network layer IP packet delivery.

In this dissertation, we address gaps left by the previous works and advances a new connection-oriented multi-hop BLE network through *Wi-BLE* which is IP- and standards-compliant.

## 1.3 Contributions and Outline

Internet of Things (IoT) is a technical megatrend in academia and industry that aims to provide connectivity to resource constrained embedded devices deployed over large areas. With the advent of Internet of Things (IoT), BLE is being used to provide wireless connectivity to resource-constrained devices in short-range applications such as smart watch, heart monitoring, and proximity services. The large deployment base of BLE equipped smartphones and wearables presents a fertile environment for further development of BLE-based IoT applications. Despite its successful commercialization, BLEs current use is limited to short-range applications which stems from its link layer design without considering a dynamic channel condition and a lack of mesh networking capability.

In general, since BLE is targeted at short-range applications, the current link layer operation of BLE is mainly designed assuming static and stable channel conditions. However, as BLE is becoming part of IoT [18][19][20] and its applications are being diversified (e.g., smart home), the link layer operation of BLE incurs significant performance degradation under dynamic channel condition. For example, there is an important link layer parameter of BLE called connection interval which affect the robustness of BLE link. For the energy efficient connection management of BLE, the parameter value should be adapted to the current channel condition but most commercial BLE devices uses fixed value of connection interval, which decreases the performance of BLE link under dynamic channel condition. This kinds of obsolete link layer design curtail BLE from taking a leading role in the market share of longer-range IoT applications such as smart homes and buildings.

BLE's lack of mesh networking capability is another issue disturbing BLE's penetration into long-range IoT applications. Therefore researchers recently started paying attention to BLE-based mesh networking to overcome its range limitation. Bluetooth Special Interest Group (SIG) formed a mesh working group in 2015 which will standardize a mesh network protocol for BLE by the middle of 2017 [14]. Even prior to standardization, some companies have released their own BLE mesh networking solutions. For example, CSR launched CSRmesh for BLE-based lighting systems [11]. Nordic Semiconductor proposed a BLE rebroadcast mesh protocol and released its implementation at Github [13]. However, their flooding-based approaches cannot exploit advantages of BLEs connection-based operation which include channel hopping, synchronous data exchange, and link level retransmission.

In this dissertation, we investigate these two issues that need to be addressed to overcome BLE's range limitation and also suggest a new application scenario based on the extended coverage of BLE network. This dissertation is organized as follows.

In Chapter 2, we investigate energy consumption and connection maintenance issues of BLE in dynamic wireless environments. We choose the connection interval of BLE as the key parameter and control variable, focusing on how to effect adaptive control in dynamic channel environments. we mathematically derive an optimal connection interval for a given channel condition and design the *CABLE* system that estimates link quality and adjusts connection interval dynamically. We evaluate the performance of *CABLE* both through extensive simulation and testbed experiments. The results show that significant performance improvement that closely approximates the optimum is achievable.

In Chapter 3, we interconnect RPL with BLE and construct a BLE-based multi-hop network. We propose a synergistic network architecture that incorporates a redesigned RPL control message broadcast, RPL parent change procedures, and RPL routing metric. This is captured by a new adaptation layer, termed *ALBER*, which enables tight coupling of RPL and BLE operations. We evaluate the performance of *RPL over BLE* through extensive experiments in an indoor testbed. We compare the performance results with that of *RPL over IEEE 802.15.4* which shows significant improvement. We also show that *RPL over BLE* is quite robust under varying link dynamics and WiFi interference. The proposed architecture significantly improves end-to-end packet delivery performance with low duty-cycle.

In Chapter 4, we propose a novel layered architecture using Wi-Fi and BLE for disaster communication. We design *Wi-BLE* to maintain a disaster network with low power consumption and also provide high data rate occasionally. For the *Wi-BLE* implementation, we design *MABLE* first to maintain the underlay BLE network with low energy consumption. We also define the format of Wi-BLE control packets (WREQ and WREP) and the two types of routing method to configure the Wi-Fi ad-hoc network as needed. Finally, we evaluate the performance of *Wi-BLE* through extensive experiments in an indoor testbed. We compare the performance results with that of Wi-Fi network using AODV routing protocol and the results show that our proposed *Wi-BLE* significantly reduces the energy consumption of nodes in our testbed.

We conclude the dissertation in Chapter 5.

# Chapter 2

# CABLE: Connection Interval Adaptation for BLE in Dynamic Wireless Environments

## 2.1 Introduction

Bluetooth (BT) is one of the most popular wireless protocols that has been widely deployed for headsets, digital cameras, smart phones, and car infotainment systems, among many others. Recently, to support applications with low energy consumption requirements, Bluetooth Low Energy (BLE) was added to the Bluetooth 4.0 specification [21][22]. Compared to classic BT, BLE has distinct features such as simple connection setup, low transmission power, and efficient data transmission. Industry has a lot of interest in BLE due to its great simplicity and energy efficiency. As a result, a wide range of commercial products are equipped with BLE due to their low power wireless communication requirement [23].

With increased BLE development, there have been efforts to make BLE more interoperable with legacy devices and systems. Bluetooth Special Interest Group (SIG) included a new logical link-layer channel dedicated to IP-based connections in the Bluetooth 4.1 specification aimed at integrating BLE as part of Internet of Things (IoT) [18]. In the recent 4.2 specification, they introduced Internet Protocol Support Profiles (IPSP) and HTTP Proxy Service (HPS) for Internet accessibility of Bluetooth smart sensors through a BLE gateway [19][20]. The recent trends indicate that BLE will play an increasingly important role in future low-power IoT applications.

The BLE standard does not specify link layer parameters, instead decoupling their values from the main standard to provide a degree of flexibility. Thus, application designers can freely determine the parameter values in accordance with application-specific requirements. An important part of link-layer BLE parameters is *connection interval* ($T_{CI}$) which controls sleep and awake periods. At every $T_{CI}$, a pair of connected BLE nodes wake up, exchange a null packet to maintain connection, engage in needed data transmission, and sleep again.

Since $T_{CI}$ affects latency and throughput, choosing an appropriate fixed value has been an important design consideration. For example, an off-the-shelf BLE device for heart rate monitoring (Polar H7) sets $T_{CI}$ to 1 second to deliver heart rate information every 2 seconds. In general, since BLE is targeted at low data rate applications, latency is the primary performance metric of interest.

However, as BLE is becoming part of IoT [18][19][20] and its applications are being diversified (e.g., smart home), we argue that using fixed $T_{CI}$, which is determined by considering latency only, is insufficient to meet application requirements. In particular, BLE is expected to support dynamic wireless environments where it faces the issue of connection maintenance and recovery subject to low power consumption constraints. For example, smart home applications [24] such as remote door locking and gas alarm detection may involve distances between BLE nodes greater than conventional WPAN applications which creates additional challenges arising from wireless link dynamics[1] as well as path loss in indoor environments.

With respect to $T_{CI}$, when it is increased, BLE saves energy due to infrequent wake-ups, while at the same time, increase of $T_{CI}$ may result in increased energy consumption due to more frequent connection losses[2]. The trade-off points toward adaptive $T_{CI}$ control that

[1]In smart home applications, when two BLE nodes are installed in different rooms, opening and closing doors incur change of link quality.

[2]This is because a BLE node loses connectivity when failing to exchange packets for a predetermined period (i.e., supervision timeout period), and a large $T_{CI}$ reduces chances to transmit packets during the supervision timeout period.

incorporates both connectivity maintenance and energy consumption to reduce energy consumption of a BLE device in dynamic wireless environments.

In this paper, we tackle the BLE connection maintenance and energy consumption problems by adaptively controlling $T_{CI}$ under dynamic channel conditions. We formulate an optimization problem to find an optimal $T_{CI}$ and design a connection interval adaptation mechanism for BLE (*CABLE*) to achieve high energy efficiency while maintaining robust connectivity. Our main contributions are summarized as follows.

- Using empirical measurements in an indoor testbed, we show that use of fixed $T_{CI}$ (currently common design choice) can result in severe performance degradation in the form of frequent connection losses and large energy consumption.
- We formulate and solve an optimization problem which finds an optimal $T_{CI}$ for energy consumption minimization subject to desired connection maintenance.
- Based on the analysis, we design *CABLE* that adapts $T_{CI}$ according to varying channel conditions, and implement *CABLE* on Linux-based real embedded devices.
- We extensively evaluate the performance of *CABLE* through testbed experiments and simulation which show that *CABLE*

reduces energy consumption up to 38% while achieving robust connectivity in dynamic channel environments.

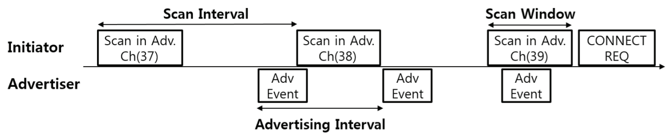
The remainder of this chapter is organized as follows. Section 3.2 describes the BLE protocol and its performance issues based on measurements in an indoor testbed. In Section 2.3, we formulate an optimization problem for $T_{CI}$ in dynamic wireless environments and analytically find an optimal solution. Section 2.4 proposes *CABLE*, and Section 3.5 evaluates its performance through extensive testbed experiments and simulation. Section 3.6 concludes this chapter.

## 2.2 Background and Problem Statement

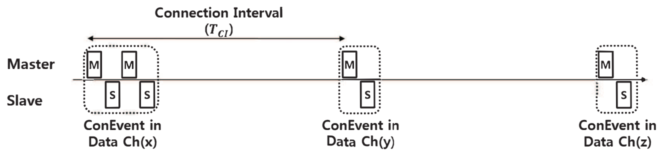
In this section, we summarize aspects of BLE relevant to the following Bluetooth 4.1 core specification. We show experiments from a BLE testbed using fixed $T_{CI}$ and evaluate performance with respect to reliability and energy consumption.

### 2.2.1 Link layer operation

Link layer operations of BLE can be divided into two parts: before and after connection establishment. Before connection establishment, a device performs *device discovery* that enables it to be aware of the presence of other nodes (scanning) or broadcast its existence to other nodes (advertising). As illustrated in Figure 2.1(a), the advertiser periodically performs an advertising event (i.e., AdvEvent) of *advertising*



(a) Operation to establish a connection (before connection)



(b) Operation between two connected devices (after connection)

Figure 2.1: BLE link layer operations.

*interval* which consists of three advertising packet transmissions: one for each of three advertising channels.

At the other hand, the initiator scans each advertising channel during a *scan window* at every *scan interval* to receive advertising packets from neighbor nodes. If the initiator receives an advertising packet from a targeted device during scanning, it immediately transmits a *connection request* packet (i.e., CONNECT REQ in Figure 2.1(a)) to establish a connection with the advertiser. Otherwise, the advertiser and the initiator repeat the advertising and scanning operations until they meet each other, which may result in significant energy consumption.

Once a connection is established, the initiator and the advertiser become master and slave, and exchange data packets through duty-cycling and channel hopping. As depicted in Figure 2.1(b), at every $T_{CI}$, the master and the slave wake up, and the master starts a *con-*

*nection event* (i.e., ConEvent) by transmitting a data packet in a data channel. The slave responds by sending an ACK or data packet. When there is no more data packet to exchange or two consecutive CRC check errors occur, they sleep, which concludes the event. At the start of each connection event, they change frequency channels based on a hopping sequence determined by the master to mitigate interference from other devices in the 2.4GHz band. The master can maintain multiple connections with multiple slaves simultaneously by scheduling *connection events* not to overlap with each other in time domain.

### 2.2.2 Connection loss due to supervision timeout

Once a pair of BLE nodes make a connection, they maintain it only when they have a valid wireless link. To check link validity, BLE allows nodes to exchange a null packet every $T_{CI}$ at the start of each connection event, even when there is no data packet to be exchanged. The two BLE nodes in connection detect connection loss when all packet exchange attempts fail during the *supervision timeout period* ($T_{ST}$). The two nodes have at least[3] $\lfloor T_{ST}/T_{CI} \rfloor$ chances to exchange null or data packets until the supervision timeout mechanism kicks in. Decreasing $T_{CI}$ enhances robustness of connection maintenance.

Connection maintenance needs to be considered carefully since it impacts battery life. Each connection loss incurs large energy con-

[3] $\lfloor x \rfloor$ is the largest integer that is not greater than $x$.

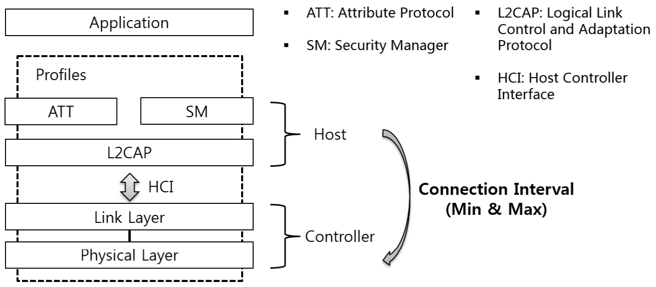


Figure 2.2: BLE protocol stack and determination of Connection Interval.

sumption by forcing the disconnected nodes to restart connection recovery which entails exchange of many control packets[4]. In this paper, given the importance of $T_{CI}$ for connection maintenance and energy consumption, we control $T_{CI}$ to maximize battery lifetime while achieving robust connection maintenance in dynamic channel environments.

### 2.2.3 BLE protocol stack and connection interval setting

The protocol stack of BLE is divided into host and controller parts (i.e., upper and lower layers) as depicted in Figure 3.5. The host part consists of upper layers (i.e., ATT, SM, and L2CAP) which are included in host devices such as mobile phone, tablet, or laptop. It acts as an interface which allows an application to use functionalities of

[4]Whenever a pair of BLE nodes establish a new connection for an application service, they initially use very small $T_{CI}$ (50∼70msec) to rapidly exchange many control packets for service discovery and encryption. The small $T_{CI}$ allows them to avoid long service time delay at the expense of large energy consumption [25].

lower layers. The controller part comprises lower layers (i.e., link and physical layers) which deal with link level operations such as device discovery, connection management and packet exchange. Since this part is generally implemented on a hardware chipset, an application cannot directly control its operations.

However, BLE provides various Host Controller Interface (HCI) commands which allow the host to manage some lower layer operations and parameters. For example, the host can set *Conn_Interval_Min ($T_{CI,min}$)* and *Conn_Interval_Max ($T_{CI,max}$)* by sending an *LE Create Connection Command* or *LE Connection Update Command* to the controller. The two parameters determine the available range of connection interval ($T_{CI}$) in the controller part (i.e., $T_{CI,min} \leq T_{CI} \leq T_{CI,max}$).

Since the BLE standard does not specify how the controller selects a $T_{CI}$ value within the given range [$T_{CI,min}$, $T_{CI,max}$], design and implementation of $T_{CI}$ selection is solely up to BLE chipset vendors. In practice, it is not guaranteed that the fixed $T_{CI}$ selected by a controller is well-suited for dynamic channel environments, which can result in significant performance degradation. Thus, application designers generally specify $T_{CI}$, by default fixed, by using the *LE Create Connection Command* with the same value for $T_{CI,min}$ and $T_{CI,max}$.

As we will see next, use of fixed $T_{CI}$ can lead to the significant waste of energy that our adaptive control *CABLE* will mitigate.

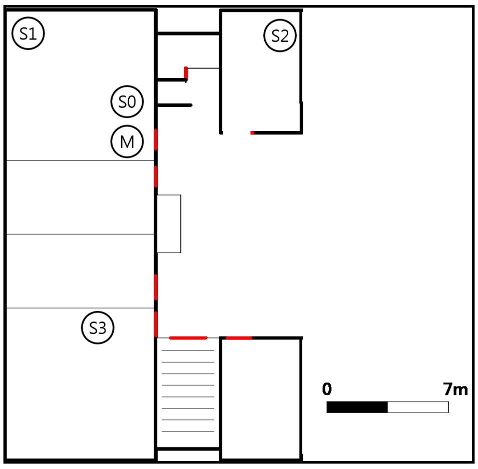
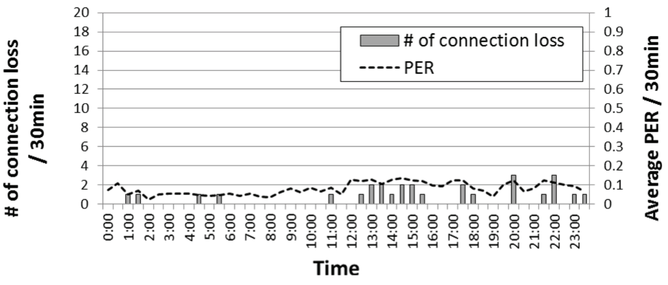


Figure 2.3: Measurement topology.

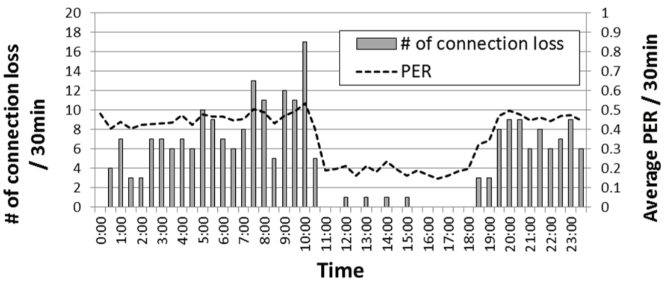
## 2.2.4 Problem of BLE with fixed connection interval

To experimentally gauge the performance degradation stemming from fixed $T_{CI}$, we configured a testbed as depicted in Figure 3.7. Four slave nodes (nodes $S0$ through $S3$) and one master node (node $M$) are deployed in an indoor office environment. Each BLE node consists of a raspberry Pi as a Linux device and a CSR 8510 as its BLE module. For each of the four node pairs, we measured the packet error rate (PER) and connection losses over 24 hours. We fixed $T_{CI}$ and $T_{ST}$ to 1.5 sec and 6 sec, respectively.

To highlight the problem, we compare the performances of link 0 (between slave $S0$ and master $M$; best case) and link 2 (between slave $S2$ and master $M$; worst case). Slave $S0$ is only 0.5m apart from master $M$ and no obstacle exists between them (i.e., they are in line-of-sight) which represents a common WPAN scenario and endows



(a) Link 0 (small distance and static channel)



(b) Link 2 (large distance and dynamic channel)

Figure 2.4: PER and the number of connection losses at 30 minute intervals for links 0 (best case) and 2 (worst case).

link 0 with the best quality among the four links. On the other hand, slave $S2$ is located much farther away from master $M$ than slave $S0$, and a metal door and walls attenuates RF signal propagation of link 2, making it the worst among the four. We emphasize that IoT applications (e.g., smart home) require BLE to provide seamless packet delivery and/or high energy efficiency over dynamic wireless links like link 2.

Figure 2.4 plots PER and the number of connection losses at 30 minute intervals for links 0 and 2. First, Figure 2.4(a) shows that for link 0, both PER and the number of connection losses stay, overall, at a low level, which is conducive to BLE performance. On the other hand, from Figure 2.4(b) we observe that the PER of link 2 is much higher than that of link 0 and also exhibits significant variability over time. We inject non-stationarity by opening the metal door from 10AM to 6PM, which results in significantly lowered PER.

During the period when the door is closed, high PER leads to supervision timeouts and connection losses, which causes link 2 to suffer from frequent connection losses. This, in turn, significantly increases energy consumption of the BLE node.

The testbed experiments show that using fixed $T_{CI}$ can lead to significant performance degradation in BLE networks. For link 2, $T_{CI}$ is too large for the high PER when the door is closed. That is, the number of opportunities for packet exchanges during $T_{ST}$ is too small to maintain the connection over the challenging link conditions, which causes frequent connection losses. Furthermore, even when link 2 is subject to low PER when the door is open, additional opportunity exists to increase $T_{CI}$ without compromising connection maintenance to reduce energy consumption. The experimental results lead us to investigate dynamically controlling $T_{CI}$ based on time-varying channel conditions to minimize energy consumption while achieving robust connectivity.

## 2.3 Connection Interval Optimization

In this section, we present a mathematical analysis to find an optimal $T_{CI}$ that minimizes the average power consumption of a BLE node for maintaining connectivity. To do this, we express the power consumption of a connected BLE node as a function of $T_{CI}$, and formulate an optimization problem to find an optimal $T_{CI}$ with given channel PER.

### 2.3.1 Problem formulation

We formulate an optimization problem to find an optimal $T_{CI}$ as

$$\begin{aligned} \underset{T_{CI}}{\text{minimize}} \quad & P(T_{CI}) \\ \text{subject to} \quad & T_{CI} \le T_{CI,max}, \\ & T_{CI} \ge T_{CI,min} \end{aligned} \tag{2.1}$$

where $T_{CI,max}$ and $T_{CI,min}$ are given by the *LE Create Connection Command* (refer to Subsection 3.2.2).

The objective function $P(T_{CI})$ consists of two parts

$$P(T_{CI}) = P_{valid}(T_{CI}) + P_{recover}(T_{CI}) \tag{2.2}$$

where $P_{valid}(T_{CI})$ is the power consumption of regular operations for a valid connection (i.e., null packet exchanges) and $P_{recover}(T_{CI})$ is that for connection recovery. Since two connected BLE nodes exchange a

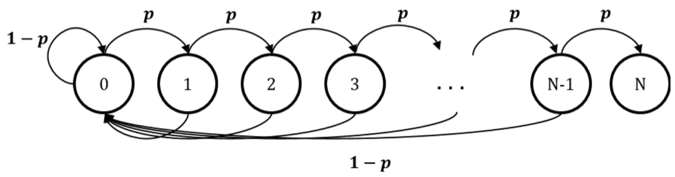


Figure 2.5: State diagram to compute the average interarrival time of supervision timeout (i.e., $ST_{avg}(T_{CI})$).

null packet at every $T_{CI}$, $P_{valid}(T_{CI})$ is given by

$$P_{valid}(T_{CI}) = \frac{E_{null}}{T_{CI}} \tag{2.3}$$

where $E_{null}$ is the energy consumption for a null packet exchange. Likewise, $P_{recover}(T_{CI})$ can be represented as

$$P_{recover}(T_{CI}) = \frac{E_{recover}}{ST_{avg}(T_{CI})} \tag{2.4}$$

where $E_{recover}$ is the energy consumption for connection recovery and $ST_{avg}(T_{CI})$ is the average inter-arrival time of supervision timeout events (i.e., connection loss). In Eqs. (2.3) and (2.4), $E_{null}$ and $E_{recover}$ are constants which can be calculated from BLE application profiles and previous related works [2][7][8][25].

$ST_{avg}(T_{CI})$ needs to be further analyzed because it is a function of $T_{CI}$. To compute $ST_{avg}(T_{CI})$, we draw a state diagram shown in Figure 2.5 where a state represents the number of consecutive null packet losses and the transition probability $p$ is the failure probability of a null packet exchange. Let $N$ be the number of chances for null packet exchanges during $T_{ST}$,

$$N = \lfloor \frac{T_{ST}}{T_{CI}} \rfloor \tag{2.5}$$

where $T_{ST}$ is predetermined by an application profile. The last state $N$ represents a supervision timeout event and $ST_{avg}(T_{CI})$ is the average time to arrive at state $N$ from state 0.

When a chain process starts from state 0, it ends up at one of two cases: arriving at state $N$ (i.e., supervision timeout after $N$ consecutive null packet losses) or returning to state 0 (i.e., at least one successful null packet exchange before the supervision timeout). We call the former *Timeout* and the latter *Return*. Letting $p_T$ and $p_R$ be the probability that *Timeout* and *Return* events occur, we have $p_T = p^N$ and $p_R = 1 - p^N$. Let $t_T$ and $t_R$ denote the average time duration until *Timeout* and *Return*. Then, $t_T = T_{ST}$ and $t_R$ can be expressed as

$$\begin{aligned} t_R &= \sum_{k=1}^{N} (kT_{CI}) \times \frac{(1-p)p^{k-1}}{p_R} \\ &= \frac{T_{CI}\{1-(N+1)p^N + Np^{N+1}\}}{(1-p^N)(1-p)}. \end{aligned} \tag{2.6}$$

The first term $kT_{CI}$ is the time duration when *Return* occurs after $k-1$ consecutive null packet losses and one successful null packet transmission, and the second term $(1-p)p^{k-1}/p_R$ is the probability of this event.

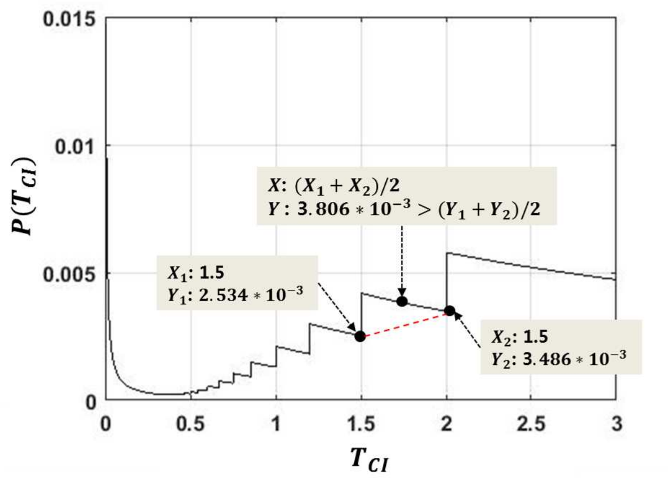


Figure 2.6: Power consumption of BLE for maintaining connectivity as a function of $T_{CI}$ (i.e., $P(T_{CI})$).

From the four parameters $p_T$, $p_R$, $t_T$, and $t_R$, we calculate $ST_{avg}(T_{CI})$ as

$$\begin{aligned} ST_{avg}(T_{CI}) &= \sum_{k=0}^{\infty} (t_T + kt_R) p_T {p_R}^k \\ &= t_T + t_R \frac{p_R}{(1 - p_R)} \\ &= T_{ST} + \frac{T_{CI}\{1 - (N+1)p^N + Np^{N+1}\}}{(1-p)p^N} \end{aligned} \tag{2.7}$$

where $k$ is the number of $Return$ events that happened before $Timeout$. Finally, $P(T_{CI})$ in Eq. (2.2) can be rewritten as

$$P(T_{CI}) = \frac{E_{null}}{T_{CI}} + \frac{E_{recover}}{T_{ST} + \frac{T_{CI}\{1-(N+1)p^N + Np^{N+1}\}}{(1-p)p^N}}. \tag{2.8}$$

Figure 2.6 plots $P(T_{CI})$ as a function of $T_{CI}$. The parameters are set as $E_{recover} = 6.27 * 10^{-2}$J, $E_{null} = 7.35 * 10^{-5}$J, $T_{ST} = 6$sec, and $p = 0.6$, based on an off-the-shelf BLE device (Polar H7) and a representative energy model for BLE [8]. The figure shows that $P(T_{CI})$ is not convex since there are two points $x_1$ and $x_2$ where $P((x_1 + x_2)/2) > (P(x_1) + P(x_2))/2$ (i.e., a counter example of convexity).

### 2.3.2 Problem solution

To address the non-convexity of $P(T_{CI})$, we convert the original problem to a convex optimization problem using the following proposition.

**Proposition 1** *The optimal $T_{CI}$ ($T_{CI,opt}$) that minimizes $P(T_{CI})$ is included in the set $\mathbf{T}_{CI,opt}$ given by*

$$\mathbf{T}_{CI,opt} = \left\{ x \,\middle|\, x = \frac{T_{ST}}{n}, \text{ where } n \text{ is any positive integer} \right\}.$$

**Proof of Prop. 1** *Consider the range of $T_{ST}/(n+1) < T_{CI} \leq T_{ST}/n$ for any positive integer $n$, where $N(= \lfloor T_{ST}/T_{CI} \rfloor)$ is fixed to $n$. $ST_{avg}(T_{CI})$ becomes a monotonically increasing function of $T_{CI}$, and $P(T_{CI})$ becomes a monotonically decreasing function of $T_{CI}$. This makes the optimal $T_{CI}$ in the given range become $T_{ST}/n$ (i.e., maximum value).*

Using the proposition, we reformulate the original problem as an optimization problem of a positive integer variable $n(= T_{ST}/T_{CI,opt})$ as follows:

$$\begin{aligned} \underset{n}{\text{minimize}} \quad & P(n) \\ \text{subject to} \quad & \lfloor \frac{T_{ST}}{T_{CI,max}} \rfloor \leq n \leq \lfloor \frac{T_{ST}}{T_{CI,min}} \rfloor . \end{aligned} \tag{2.9}$$

By replacing $T_{CI}$ of $P(T_{CI})$ with $T_{ST}/n$, we arrive at the objective function $P(n)$

$$P\left(n\right) = \frac{E_{null}}{T_{ST}} \times n + \frac{E_{recover}}{T_{ST}} \times \frac{(1-p)np^n}{1-p^n} \tag{2.10}$$

The second derivative of $P(n)$ is given by

$$\begin{aligned} P''\left(n\right) = & \frac{E_{recover}}{T_{ST}} \times \\ & \frac{\left(1-p\right)p^n \left\{ 2\ln p \left(1-p^n\right) + n(\ln p)^2 \left(1+p^n\right) \right\}}{\left(1-p^n\right)^3} . \end{aligned} \tag{2.11}$$

Note that $P''(n) > 0$ since $n > 0$ and $0 \leq p \leq 1$, which implies that $P(n)$ is convex in $n$. Thus, the reformulated optimization problem is a convex optimization problem with an integer constraint.

Using convexity of $P(n)$, we find an optimal $n$, denoted as $n_{opt}$, efficiently by increasing $n$ by one from $\lfloor T_{ST}/T_{CI,max} \rfloor$ until $P(n) < P(n+1)$ or $n = \lfloor T_{ST}/T_{CI,min} \rfloor$ which yields the optimal solution for the original problem, $T_{CI,opt} = T_{ST}/n_{opt}$.

In practice, $T_{CI,opt}$ needs to be further adjusted according to the constraint given by the BLE specification. That is, $T_{CI}$ should be a multiple of 1.25msec. To this end, we get an approximate optimal value of $T_{CI,opt}$ as a multiple of 1.25msec that is closest to $T_{ST}/n_{opt}$.

That is,

$$T_{CI,opt} = 1.25\text{msec} \times \lfloor \frac{\mathrm{T_{ST}/n_{opt}}}{1.25\text{msec}} \rfloor. \tag{2.12}$$

Even though $T_{CI,opt} \neq T_{ST}/n_{opt}$, we will show that performance degradation due to this difference is negligible in Section 3.5.

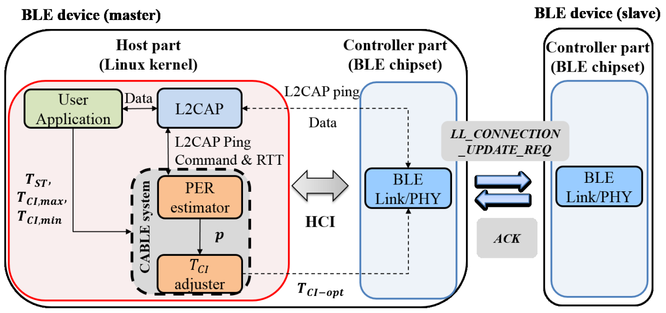


Figure 2.7: Overview of the *CABLE* system.

## 2.4 CABLE System Design

In this section, we describe our *CABLE* system that estimates link quality (i.e., PER $p$) and obtains $T_{CI,opt}$ to minimize energy consumption while maintaining connectivity. Fig. 2.7 depicts an overview of the *CABLE* system. Reflecting that a master node generally has higher computing power and battery capacity than a slave node, we design the master to take the overall role of *CABLE* operation. Specifically, the master calculates $T_{CI,opt}$ of a connection based on the estimated PER and updates its $T_{CI}$ by sending an *LL_CONNECTION_UPDATE_REQ* packet to the slave. We implement *CABLE* by modifying the host part of the master node since the controller part is generally implemented in hardware chipset firmware that is difficult to modify. Our *CABLE* system consists of two parts: *PER estimator* and $T_{CI}$ *adjuster*. *PER estimator* estimates the PER $p$ of the link between two connected nodes and delivers it to $T_{CI}$ *adjuster*. This allows $T_{CI}$ *adjuster* to calculate $T_{CI,opt}$. The other parameters for $T_{CI,opt}$ calculation (i.e.,

$T_{ST}$, $T_{CI,min}$, and $T_{CI,max}$) are given by the user application[5].

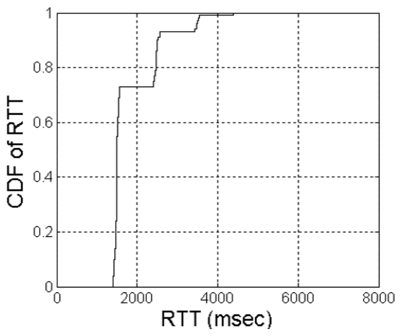
The following two subsections describe the detailed procedures of our *PER estimator* and $T_{CI}$ *adjuster*.

### 2.4.1 PER estimator

Since the controller part does not provide any information (e.g., retransmission count) to the host part for PER estimation, we design a new PER estimation scheme for the host part. Our intuition is to utilize round trip time (RTT) of L2CAP ping packets for link quality estimation which can be measured at the host.

To investigate the feasibility of using RTT for PER estimation, we use Figure 3.4 that plots the CDF of RTT values of L2CAP ping packets under two different link conditions (LOS and non-LOS links with the distance 10m) when $T_{CI} = 1$sec. Figure 3.4 shows that RTT values over the LOS link are much smaller than those over the non-LOS (NLOS) link. This indicates that RTT captures link quality and may be effective for PER estimation. We also find out that RTT values are quantized according to the step size $T_{CI}$ which comes from the BLE's retransmission strategy. That is, when a BLE node fails to transmit a packet, it closes its current connection event and delays its

[5]In the case of $T_{ST}$ and $T_{CI,max}$, *CABLE* should adopt these values given from the application because they are related with application service requirements (i.e., connection loss detection delay and maximum tolerable latency). On the other hand, we let *CABLE* overwrite the $T_{CI,min}$ value given from the application also with the minimum allowed value of 7.5msec for $T_{CI}$ adaptation. Note that such a setting to a smaller $T_{CI,min}$ than the given one from the application helps to provide better QoS than the required.


(a) LOS link with 10m distance

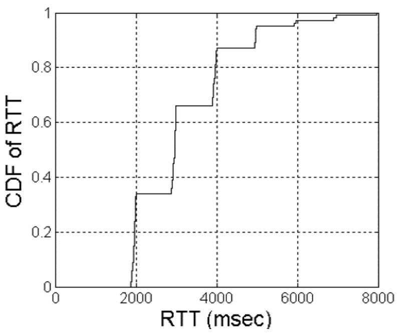

(b) NLOS link with 10m distance

Figure 2.8: CDF of Logical Link layer ping's RTT with $T_{CI}$ = 1sec.

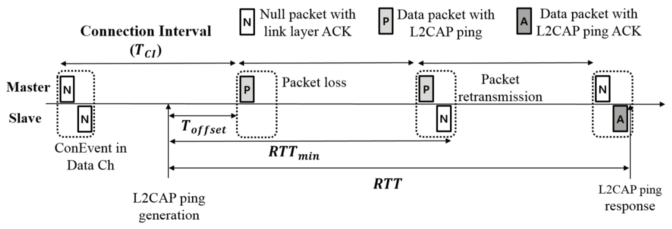


Figure 2.9: PER estimation based on L2CAP ping's RTT.

retransmission attempt until the next connection event starts. This leads to retransmission being delayed by one $T_{CI}$.

Figure 2.9 illustrates quantized RTT by describing link-layer retransmission procedures of BLE. When the host part of the master node generates an L2CAP ping packet, it delivers the packet to the link layer's Tx queue and transmits it at the start of the next connection event (i.e., after a delay of $T_{offset} < T_{CI}$). If the transmission is successful, the master node receives an ACK after one $T_{CI}$, which results in the minimum RTT, denoted as $RTT_{min}$ of $(T_{offset} + T_{CI})$. In case of failure, it retransmits the ping packet at the next connection event which increases RTT to $T_{offset} + 2T_{CI}$.

Based on these observations, we can estimate the number of link layer transmissions required for a ping packet, $N_{retx}$, using each measured RTT value as

$$N_{retx} = \left\lfloor \frac{RTT}{T_{CI}} \right\rfloor \tag{2.13}$$

and we estimate PER $p$ as

$$p = \frac{N_{retx} - 1}{N_{retx}} \tag{2.14}$$

where $(N_{retx} - 1)$ indicates the number of erroneous ping packet transmissions. We design our *PER estimator* to generate an L2CAP ping packet over a period of $T_{ping}$. After each ping packet exchange, *PER estimator* updates $p$ by using a moving average filter [6] and provides the estimate to $T_{CI}$ *adjuster* to optimize $T_{CI}$.

There is a trade-off relationship between ping packet overhead and estimation accuracy in our PER estimation. We find that estimation overhead is marginal compared to overhead from the BLE's default connection management mechanism. Since a connected node sends a ping packet (size 21 bytes) instead of a null packet (10 bytes), each ping exchange incurs 176 $\mu$sec additional radio-on time. In our testbed experiments, this results in only an increase of 0.00176% in duty cycle since we empirically set $T_{ping}$ to 10 seconds. Our performance evalu-

[6] A single PER value reflects link quality poorly, not only because a wireless link fluctuates over time but also BLE continuously changes its frequency channel according to the hopping pattern.

ation shows that this $T_{ping}$ value and our PER estimation scheme are suitable for *CABLE*.

### 2.4.2 $T_{CI}$ adjuster

**Algorithm 1** $T_{CI}$ adaptation

```
 1: procedure CIopt(p)
 2:     With PER value $p$ given by PER estimator, and constant system parameters $T_{ST}$, $T_{CI,min}$, and $T_{CI,max}$
 3:     for $n = \lfloor T_{ST}/T_{CI,max} \rfloor \to \lfloor T_{ST}/T_{CI,min} \rfloor$ do
 4:         if $P(n) < P(n+1)$ then
 5:             break
            end if
        end for
 6:     $n_{opt} \leftarrow n$
 7:     $T_{CI,opt} \leftarrow 1.25\text{msec} \times \lfloor \frac{T_{ST}/n_{opt}}{1.25\text{msec}} \rfloor$
 8:     return $T_{CI,opt}$
    end procedure
 9: procedure CIupdate($p_{new}$)
10:     if $|p_{new} - p| > p_{th}$ then
11:         $T_{CI,opt} \leftarrow CIopt(p_{new})$
12:         Update $T_{CI}$ with LE Connection Update Command
13:         $p \leftarrow p_{new}$
        end if
    end procedure
```

Our $T_{CI}$ *adjuster* updates $T_{CI}$ based on the PER $p$ given by *PER estimator* as presented in Algorithm 1. The procedure ***CIopt()*** in Algorithm 1 is designed to find $T_{CI,opt}$ within the range ($T_{CI,min} \leq T_{CI} \leq T_{CI,max}$) which is given by *LE Create Connection Command* from an application profile. It gets an optimal $T_{CI}$ for the PER value $p$ by using the convexity of $P(n)$ in Eq. (2.10). Specifically, it increases $n$ by one from $\lfloor T_{ST}/T_{CI,max} \rfloor$ until $P(n) < P(n+1)$ or $n = \lfloor T_{ST}/T_{CI,min} \rfloor$, and uses the final output $n$ for $n_{opt}$. Then, it chooses a multiple of 1.25msec which is closest to $T_{ST}/n_{opt}$ as the optimal value $T_{CI,opt}$.

$T_{CI}$ *adjuster* executes the procedure ***CIupdate()*** whenever a new PER value $p_{new}$ is reported by *PER estimator*. Our intuition behind the design of ***CIupdate()*** is that we need to prevent too frequent $T_{CI}$ updates since each $T_{CI}$ update incurs several control packet exchanges and consumes additional energy. To this end, ***CIupdate()*** updates $T_{CI,opt}$ only when the gap between the new PER $p_{new}$ (reported by *PER estimator*) and the current PER $p$ (used by $T_{CI}$ *adjuster* in the previous $T_{CI}$ update) is larger than a threshold $p_{th}$. When it determines to update $T_{CI}$, it calculates the optimal value $T_{CI,opt}$ by executing ***CIopt()***. Finally, it updates $T_{CI}$ of the ongoing connection by sending an *LE Connection Update Command*[7] to the controller part and replaces $p$ with $p_{new}$.

## 2.5 Performance Evaluation

We evaluate *CABLE* through both real-world experiments and simulation. In simulation, we assume an idealized case where we can modify the controller part of BLE which allows *CABLE* to obtain PER

[7]$T_{CI}$ adjuster sets both $T_{CI,max}$ and $T_{CI,min}$ in *LE Connection Update Command* to $T_{CI,opt}$, which forces the controller part to replace the current $T_{CI}$ with the new $T_{CI,opt}$ (refer to Section 3.2.2). Then, the controller part starts *Connection Update Procedure* with its slave node to apply the updated $T_{CI}$.

Table 2.1: Parameters for BLE operation

| | | | |
|---|---|---|---|
| $T_{AI}$ | 30msec | $T_{CI,min}$ | 7.5msec |
| $T_{SI}$ | 60msec | $T_{CI,max}$ | 2sec |
| $T_{SW}$ | 30msec | $T_{ST}$ | 6sec |

information without additional ping packet exchanges. The results under two different channel scenarios show that *CABLE* outperforms state-of-the-art BLE link layer protocols with fixed $T_{CI}$. In experimental benchmarking, we evaluate *CABLE* on real embedded devices in an indoor testbed environment. The results show that our PER estimation scheme provides accurate estimation of link quality with low overhead and *CABLE* outperforms the current BLE in terms of energy saving.

Table 2.1 shows the parameters for BLE operation used in our performance evaluation. There are three operation parameters for device discovery: advertising interval $T_{AI}$, scan interval $T_{SI}$, and scan window $T_{SW}$. These parameter values come from the BLE application profile defined by Bluetooth SIG [25]. We determine the parameter values for connection maintenance (i.e. $T_{CI,min}$, $T_{CI,max}$, and $T_{ST}$) based on an off-the-shelf BLE device (Polar H7). Lastly, we set $p_{th}$ to 0.1 based on empirical testing.

### 2.5.1 Simulation results

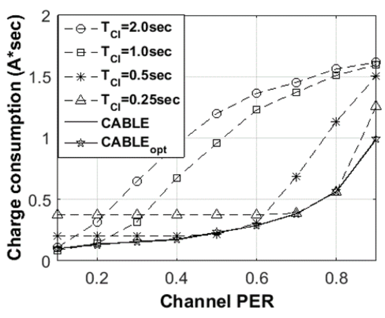
In our simulation, we assume that the BLE controller part can be modified and implement *CABLE* on the controller part by using a

C++ based simulator. In this ideal case, *CABLE* updates PER information upon every (null) packet exchange at link layer, without additional overhead for link estimation (i.e., L2CAP ping exchange). Thus, we remove our PER estimation scheme in this case. This scenario allows us to gauge the performance of *CABLE* in the event that BLE chipset vendors adopt *CABLE* for their BLE chipsets[8].

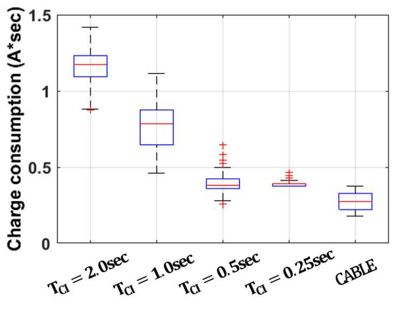
We consider two BLE nodes that maintain a BLE connection in the context of a smart home application. Energy consumption is evaluated by charge consumption of the slave node's BLE chipset, assuming that the master node is a wall-powered device (e.g., BLE home gateway). We test *CABLE*'s performance in two different channel scenarios for an hour. The results are shown in Figure 2.10. First, Figure 2.10(a) considers the case where all 40 BLE channels have the same and static PER, which shows the effect of channel PER on the performance. Second, we use a real-world channel trace to evaluate performance under dynamic channel conditions in Figs. 2.10(b) through (d). To obtain the channel trace, we capture all BLE channels in our indoor testbed by using *nRF-Sniffer* of Nordic Semiconductor [26].

For comparison, we evaluate the performance of the conventional BLE protocol using fixed connection intervals 0.25sec, 0.5sec, 1.0sec, and 2.0sec. With this performance comparison, we clarify the necessity of adaptive connection interval control and the superiority of

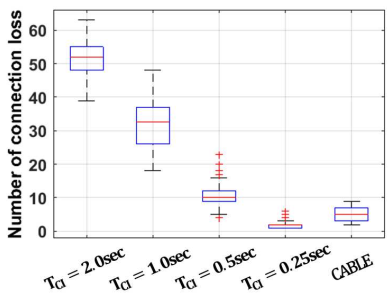
[8]Based on our reverse-engineering with a few BLE chipset, we confirmed that most of current off-the-shelf BLE chipset do not adapt $T_{CI}$ according to channel condition. Thus, we foresee this work as a crucial ground in asking BLE chipset vendors to adopt $T_{CI}$ adaptation mechanism like *CABLE*.



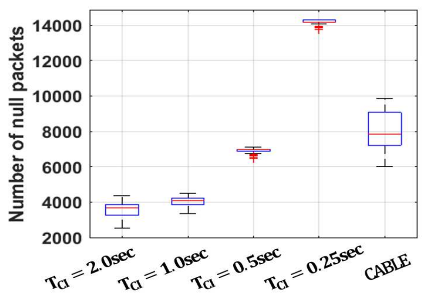
(a) Charge consumption with static channel



(b) Charge consumption with real channel trace



(c) Connection losses with real channel trace



(d) Null packets with real channel trace

Figure 2.10: Simulation results for two different channel scenarios; figure (a) for the same and static channel PER scenario, and figures (b)$_{\sim}$(d) for a real channel trace scenario.

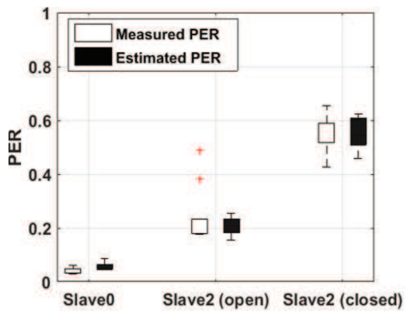
our *CABLE*. We evaluate a variant of *CABLE*, $CABLE_{opt}$, in Figure 2.10 that finds $T_{CI,opt}$ with exhaustive search for all available $T_{CI}$ values. $CABLE_{opt}$ serves as a reference point for what is, in principle, achievable.

Figure 2.10(a) plots the charge consumption of the BLE chipset at the slave node as a function of channel PER when all BLE channels have the same and static channel PER. First, we observe that charge consumption increases with channel PER in all cases. This is because high channel PER incurs frequent supervision timeouts which increase control overhead for connection recovery. By comparing the four cases of fixed $T_{CI}$, we find that as the channel PER changes, the value of fixed $T_{CI}$ that achieves the minimum charge consumption also changes. For example, $T_{CI} = 1.0$sec outperforms the other cases when $PER = 0.2$, while $T_{CI} = 0.25$sec provides the best performance when $PER = 0.7$. The results show that BLE performance may be significantly improved by adapting $T_{CI}$ to time-varying channel conditions.

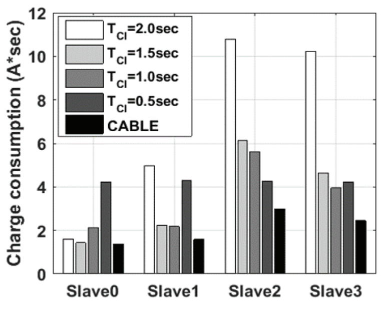
The performance results of *CABLE* show it is able to respond to varying channel conditions using the estimated PER. Adaptive control of $T_{CI}$ achieves significant performance improvement which nearly matches that of $CABLE_{opt}$. This reveals that our optimization algorithm which has low computational overhead and uses multiples of 1.25msec in the discretized approximation is able to find a proper $T_{CI,opt}$ indeed without significant performance degradation. In terms of complexity, the simulation results also show that our *CABLE* reduces the number of iterations for optimal point searching in the range 87.8-99.9% over $CABLE_{opt}$ using exhaustive search. This reduced complexity helps *CABLE* to rapidly adapt $T_{CI}$ according to dynamic channel conditions.

Figure 2.10(b) through (d) show that *CABLE* outperforms the current BLE with respect to various performance metrics for the four fixed $T_{CI}$ values under a real-world channel trace. Figure 2.10(b) plots charge consumption and shows that *CABLE* reduces average power consumption in the range 27–73% over fixed $T_{CI}$ cases. Figure 2.10(c) and (d) show the trade-off relation between connection losses and null packet exchanges. As the fixed $T_{CI}$ value decreases, the number of connection losses decreases, which means that the BLE nodes can reduce energy consumption for connection recovery. On the other hand, the number of null packets exchanged increases as $T_{CI}$ decreases, which increases the energy consumption for null packet exchanges. The trade-off relation between connection losses and null packet exchanges indicates that the performance gain of *CABLE* comes from $T_{CI}$ adaptation.

For real-world experiments, we implemented *CABLE* on real embedded devices using the Raspberry Pi with Linux kernel 3.17 and CSR 8510 for the BLE module. Since we cannot modify the controller part of the BLE chipset CSR 8510, we implemented *CABLE*—both *PER estimator* and $T_{CI}$ *adjuster*—on the host part of BLE in the Linux kernel. Thus the experimental results incorporate the effect of PER estimation scheme on *CABLE*'s performance. We configured the indoor testbed environment with one master node and four slave nodes as shown in Figure 3.7. We measured charge consumption of each slave node's BLE chipset to evaluate energy consumption performance. Each slave node logs its BLE operations and we use the log file after an experiment to calculate energy consumption by using the



(a) Performance of PER estimation



(b) Charge consumption of each slave node

Figure 2.11: Experimental results from the indoor office testbed.

*BLEeMod-Energy modeling library* [8].

## 2.5.2 Experimental results

First, we investigate the performance of our *PER estimator*. To evaluate PER estimation that uses L2CAP ping's RTT, we compare the *estimated* PER results with those *measured* by *nRF-Sniffer*. Figure 2.11(a) shows the measured and estimated PERs of slaves 0 and 2 from Figure 3.7. We intentionally change the channel condition between the master and slave 2 by opening/closing the metal door as discussed in Section 2.2.4. The results show that accuracy of *PER estimator* closely matches PER from sniffer measurements. The mean gap between measured and estimated PERs is 0.016. Accurate PER estimation in dynamic BLE channels is an important building block for effective $T_{CI}$ control.

Second, we let each slave node maintain a connection with the master for 24 hours and measure total charge consumption of each slave's BLE chipset. Figure 2.11(b) plots charge consumption of the four fixed $T_{CI}$ cases and $CABLE$. In the case of fixed $T_{CI}$, the best $T_{CI}$ value (i.e., minimum energy consumption) for each slave is different depending on its channel condition. As expected, the best $T_{CI}$ value decreases as link quality becomes worse. The results show that when $T_{CI}$ is not suitable for a given link condition for connection maintenance, BLE incurs large energy consumption. In contrast, $CABLE$ achieves minimum energy consumption for all the four slave nodes. Its performance gain over the best fixed $T_{CI}$ case increases as link condition becomes worse and unstable. $CABLE$ reduces the charge consumption in the range 2.3–38% compared to fixed $T_{CI}$. Overall, the experimental results show the effectiveness of $CABLE$ in dynamic real-world channel environments, its performance gains over current BLE becoming more pronounced with increased channel variability.

## 2.6 Summary

In this paper, we investigated energy consumption and connection maintenance issues of BLE in dynamic wireless environments. We chose the connection interval of BLE as the key parameter and control variable, focusing on how to effect adaptive control in dynamic channel environments. Through testbed experiments, we established that fixed values assigned to connection interval by legacy BLE systems significantly impacts connection maintenance and energy consumption. To exploit the potential performance improvements achievable through adaptive control, we mathematically derived an optimal connection interval for a given channel condition and designed the *CABLE* system that estimates link quality and adjusts connection interval dynamically. We evaluated the performance of *CABLE* both through extensive simulation and testbed experiments. The simulation results show that if *CABLE* is implemented at the BLE controller part in firmware, significant performance gains are achievable using traces from real-world wireless channels. The experimental results show that even when *CABLE* is implemented on the host part due to the proprietary nature of today's BLE controllers, significant performance improvement that closely approximates the optimum is achievable.

# Chapter 3

# A Synergistic Architecture for RPL over BLE

## 3.1 Introduction

Internet of Things (IoT) is a technical megatrend in academia and industry that aims to provide Internet connectivity to resource constrained embedded devices deployed over large areas. When combined with recently advanced technologies such as low power and lossy network (LLN), big data analysis, low cost sensing, and machine learning, IoT has the potential to facilitate a variety of useful applications that impact industry and global markets.

There have been various standardization efforts by IEEE, IETF, ZigBee aliance, and Bluetooth Special Interest Group (SIG) to develop protocols and application profiles for IoT systems. IPv6 routing protocol for LLN, termed RPL [27], is one of such efforts. To support upcoming smart grids and many other IoT applications, RPL constructs a multi-hop LLN with a tree-like routing topology called destination-oriented directed acyclic graph (DODAG), and enables bi-directional IPv6 communication between resource constrained embedded devices.

At the link layer, Bluetooth Low Energy (BLE) and IEEE 802.15.4 are two popular low power and low cost wireless protocols. To date, research (mostly wireless sensor networks) has focused on developing various communication and network techniques on top of IEEE 802.15.4 [28][29][30][31][32][33][34][35]. Recently, IEEE 802.15.4 started to deliver IPv6 packets using 6LoWPAN as an adaptation layer, which provides interoperability with the larger Internet. As a result, IEEE 802.15.4 is widely used in RPL-based multi-hop IoT networks (e.g., Cisco's CG-Mesh network for smart grid [36]) as an underlying link layer protocol. On the other hand, BLE was designed to support single hop range applications which include wearable technologies such as smart watches and some applications in the health care IT.

BLE has not been a focus of wireless sensor network research for multi-hop communication, in contrast to other link layer technologies such as IEEE 802.15.4. In this paper, we aim to show that BLE multi-hop communication can be valuable for several reasons. First, compared to IEEE 802.15.4 [37], BLE provides four times higher data rate (250 kbps vs. 1 Mbps) as well as lower power consumption for transmission and reception. It has the capability of avoiding wireless interference from other devices by exploiting frequency hopping. Second, the synchronous characteristic of its MAC protocol can improve data delivery performance over asynchronous MACs such as ContikiMAC and BoX-MAC, which are widely used for IEEE 802.15.4-based multi-hop networks. Third, from a practical point of view, many of today's smartphones are equipped with BLE which facilitates legacy compatible deployment, accessibility and usability [23].

The Bluetooth Special Interest Group (SIG) defined a new logical link layer channel dedicated to IP-based connections in the Bluetooth 4.1 specification [18]. This specification allows a BLE node to perform multiple roles (master and slave) simultaneously which enables a BLE-based multi-hop network with hierarchical tree topology. In the most recent 4.2 specification, Internet Protocol Support Profiles (IPSP) and HTTP Proxy Service (HPS) have been standardized for the Internet accessibility of Bluetooth smart sensors through a BLE gateway [19][20]. In addition, IETF recently standardized 6LoWPAN for BLE, which enables BLE to deliver IPv6 packets [38].

As part of these trends, in this paper, we consider a protocol stack that aims to run RPL of the IETF standard over BLE to construct BLE-based multi-hop networks. Our motivation is that, when considering the current status and strength of BLE, mounting a standard routing protocol such as RPL on top of BLE may greatly improve its usability and deliver performance on par with, or better than, IEEE 802.15.4 in large scale applications.

The contributions of this paper are four-fold.

- We interconnect standard routing protocol RPL completely with BLE as a new approach. We propose a synergistic architecture for *RPL over BLE* that constructs an energy-efficient and reliable multi-hop network through connection-less broadcast of RPL control messages and connection-oriented data delivery.

- We design an adaptation layer between BLE and RPL, termed *ALBER*, which effects energy-efficient delivery of RPL control messages, connection-aware parent change, and calculation of BLE-specific routing metric for RPL. We implement the full protocol stack of *RPL over BLE* including *ALBER* in a Linux kernel.

- In an indoor multi-hop LLN testbed, we experimentally evaluate *ALBER*. We show that it enables RPL to run on top of BLE without losing each protocol's strengths, as a result of which *RPL over BLE* significantly outperforms *RPL over IEEE 802.15.4*.

- We experimentally investigate the effect of varying connection interval on the performance of *RPL over BLE* which reveals that connection interval control has a significant effect on *RPL over BLE*'s performance.

The rest of this chapter is organized as follows. Section 3.2 provides background on RPL, BLE, and 6LoWPAN for BLE. Section 3.3 presents our design of *RPL over BLE* including *ALBER*. Section 3.4

describes its implementation details. In Section 3.5, we evaluate extensively *RPL over BLE*'s performance in an indoor testbed and compare the results with that of *RPL over IEEE 802.15.4*. Section 3.6 summarizes this work.

## 3.2 Background

### 3.2.1 RPL operation

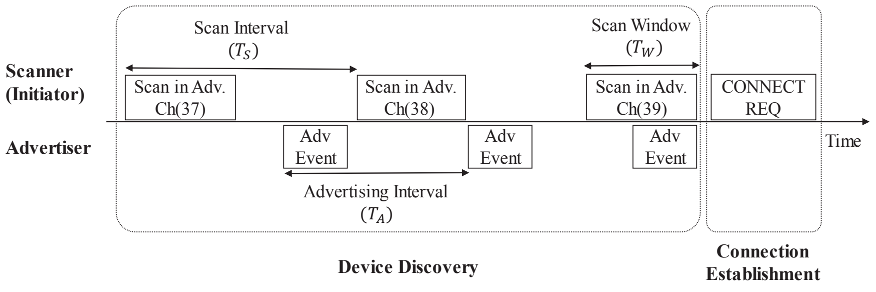
In this part, we describe ContikiRPL, which is the default RPL implementation in Contiki. RPL defines $RANK$ to represent the routing distance from a node to the root. By default, each node in ContikiRPL uses the end-to-end expected transmission count ($ETX$) toward the root as its own $RANK$. Let $ETX(n, p_n)$ represent the $ETX$ from node $n$ to its parent candidate $p_n$. Then node $n$ calculates its routing metric for each parent candidate as $R(p_n) = RANK(p_n) + ETX(n, p_n)$. RPL constructs a DODAG by designing each node to select a parent node that has the smallest routing metric among its parent candidate nodes.

Each node broadcasts the routing information including its $RANK$ using DODAG information object (DIO) messages. To achieve a balance between control overhead and fast recovery, RPL triggers DIO message transmissions based on the $TrickleTimer$ [12] which doubles the broadcast period after every DIO transmission and re-initializes it to the minimum value when route inconsistency is detected. Each node recognizes its neighbor nodes and their *RANKs* from received DIO messages. RPL constructs downward routes simply as the reverse of upward ones. To this end, each node periodically transmits destination advertisement object (DAO) messages toward the root through its upward route.

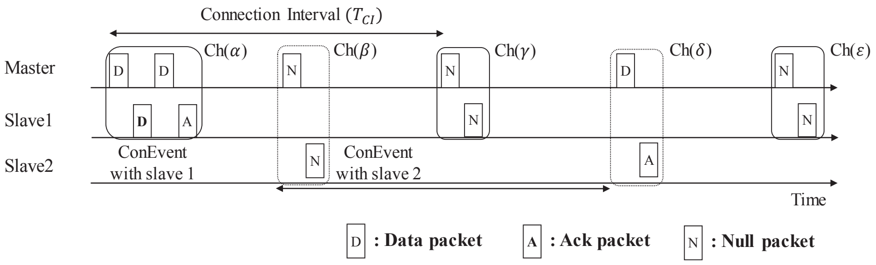
### 3.2.2 BLE link layer operation

BLE has two types of channels: 3 advertising channels and 37 data channels. Advertising channels support connection-less message exchanges through advertising and scanning as depicted in Figure 3.1(a), where the advertiser and the scanner denote a sender and a receiver, respectively. The advertiser performs an advertising event (i.e., AdvEvent in Figure 3.1(a)) periodically with advertising interval $T_A$, which comprises three message transmissions, i.e., one for each of the three advertising channels. On the other hand, the scanner tries to receive advertised packets by waking up periodically with scan interval $T_S$ and scanning an advertising channel during a scan window $T_W$. It scans a different advertising channel at each wake-up.

Based on connection-less message exchanges, BLE performs device discovery and connection establishment procedures in advertising channels. Specifically, each BLE node notifies its presence to its neighbor nodes by advertising control packets. Once a node (scanner) willing to make a connection with the advertiser detects its presence by scanning, it establishes a connection with it by sending a *Connec-*



(a) Connection-less operation on advertising channels



(b) Connection-oriented operation on data channels

Figure 3.1: Link layer operation of BLE in each channel type.

*tion Request* packet (i.e., CONNECT REQ in Figure 3.1(a)). After the two nodes established a connection, each of them becomes a master or slave node according to its role in device discovery (i.e., scanner or advertiser, respectively). Then, they exchange data packets over data channels.

There are two advantages of using data channels rather than advertising channels for data communication. First, the two nodes in connection are synchronized in time domain and share a periodic wake-up schedule (i.e., connection interval $T_{CI}$ in Figure 3.1(b)). Every $T_{CI}$ they start a connection event (ConEvent) and exchange null packets to check validity of link connection (shown as 'ConEvent with slave 2' in the figure). If they have data packets for transmission, they exchange them instead of null packets. After the packet exchange is finished, the two nodes close ConEvent and sleep together. Owing to time synchronization, a parent node (master) can schedule its children nodes (slaves) to transmit without overlapping with each other in time domain, thus resulting in contention-free data transmission.

Second, for efficient use of 37 data channels, a parent node and its connected children nodes share a *pseudo-random* hopping sequence that helps to avoid collision with other parent-children (master-slave) pairs. This frequency hopping mechanism is one of its most unique features of BLE that diffentiate it from IEEE 802.15.4. BLE adjusts the hopping sequence considering the link quality of each data channel, which mitigates interference from other systems such as WiFi by removing bad channels from the channel list for the given hopping sequence.

### 3.2.3 6LoWPAN for BLE

Since RPL is an *IPv6* routing protocol for LLNs which consist of *low cost devices*, a fundamental assumption behind this is that (long and heavy) IPv6 packets have to be delivered through a lightweight link layer. When RPL was introduced, IEEE 802.15.4 was the only lightweight link layer protocol that met this requirement. Specifically, IETF standardized an adaptation layer, named 6LoWPAN [39], which

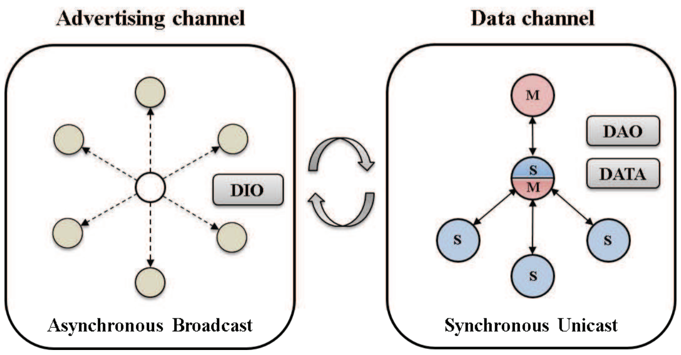


Figure 3.2: Design overview of the proposed architecture for RPL over BLE.

enables IPv6 packet transmission over IEEE 802.15.4 by using header compression and packet fragmentation. This is one of the reasons why existing RPL implementations are built on top of IEEE 802.15.4.

Recently, IETF standardized 6LoWPAN for BLE which opens the possibility of adopting another underlying link layer for RPL. Although 6LoWPAN over BLE provides almost the same functions as that over IEEE 802.15.4, it has some unique features related with BLE connection management. For instance, 6LoWPAN over BLE allows a BLE node to exchange IPv6 packets only with connected nodes and over data channels. Thus, when designing *RPL over BLE*, connection-oriented IPv6 packet exchanges over data channels are necessary, not only to improve performance but also to make it standard-compliant.

## 3.3 Design of RPL over BLE

Our work focuses on designing a feasible BLE-based multi-hop mesh network which has great compatibility, extendability, and applicability as part of IoT. To make it IPv6- and standard-compliant, we use standard IP protocols such as RPL and 6LoWPAN for BLE. Our design aims to run RPL over BLE without losing each one's strengths that are RPL's route management and BLE's energy-efficient and reliable packet delivery. To achieve this goal, we consider a new architecture for *RPL over BLE* that utilizes synergistic effects between RPL and BLE, and design *ALBER* to support this architecture. In this section, we describe design details of our proposal.

### 3.3.1 Synergistic Network Architecture for RPL over BLE

We propose a new network architecture for *RPL over BLE*, which exploits both connection-less and connection-oriented packet exchanges provided by BLE. To this end, we first analyze *pros* and *cons* of using each of the two channel types in BLE and design *RPL over BLE* to use each of them for transmission of different types of packets. The data packet exchange over data channels, which needs to operate with 6LoWPAN, consumes low energy and provides high reliability owing to synchronous operation and channel hopping. However, BLE does not allow a node pair to use data channels before they are connected and synchronized in time domain. This leads BLE to use advertising channels for device discovery and connection establishment procedures that can be performed between two nodes without connection.

We combine channel usage features of BLE with neighbor discovery and parent selection functionalities of RPL, as depicted in Figure 3.2. That is, a node broadcasts DIO messages over advertising channels, which enables its unconnected neighbor nodes to obtain its routing information. Each node selects a parent node among its parent candidates using received DIOs from its neighbor nodes, and then establishes a BLE connection with a parent node through advertising channels. Then, it exchanges data packets and DAOs with the connected parent node (master) over data channels.

Even after a node joins the network through a valid parent node, it scans advertising channels every scan interval to update its parent candidate set by using received DIOs. Furthermore, it performs periodic wake-ups in data channels and exchanges data packets with its parent node or a child node. This design choice allows resource constrained nodes to benefit from both RPL and BLE. That is, they exchange data packets reliably and energy efficiently using BLE, while constructing and recovering multi-hop routing topology using RPL.

Lastly, this architecture completely avoids collision between control and data packets because their transmissions are separated in the frequency domain, which provides contention-free data delivery even when the routing topology dynamically changes.

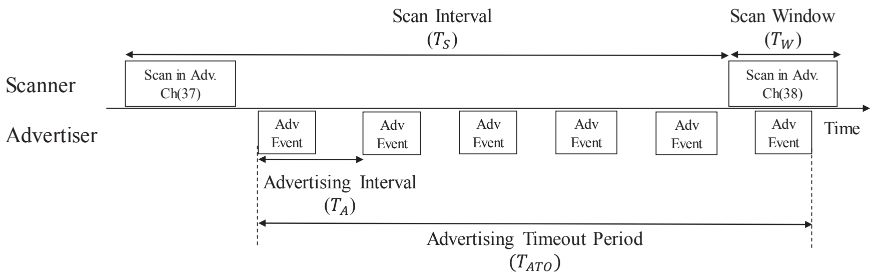


Figure 3.3: Exchange of RPL control messages over advertising channels with low power listening.

### 3.3.2 DIO broadcast over advertising channels

In the BLE standard specification, unlike data channels, advertising channels deliver packets in an asynchronous manner, which is highly affected by the four operation parameters: scan interval ($T_S$), scan window ($T_W$), advertising timeout period ($T_{ATO}$)[1], and advertising interval ($T_A$). However, when using advertising channels with the default parameter settings, a BLE node suffers from low packet delivery ratio and low energy efficiency. We need to improve the performance by tuning these parameters for routing topology management, given that DIO messages that contain important routing information are delivered through advertising channels.

We use a low power listening (LPL) protocol [29] with proper settings of the above four parameters, as depicted in Figure 3.3. For reliable delivery in advertising channels, a scanner detects at least one

[1]For DIO message broadcasting, a BLE node repeats advertising events during an advertising timeout period. Even though BLE specification does not specify this period, we newly define it for our design, given that a BLE chipset allows its upper layer to start or stop sending advertising packets.

advertising packet during a scanning procedure, which runs with the two constraints: $T_{ATO} \geq T_S$ and $T_A \leq T_W$. In addition, the transmission overhead needs to be minimized to reduce energy consumption for sending advertising packets. To this end, we set $T_{ATO} = T_S$ and $T_A = T_W$.

With this configuration, we further minimize energy consumption of a node in *advertising* channels by parameter optimization. The average power consumption of a BLE node can be expressed as

$$P = \left(\frac{T_W}{T_S}\right) I_s V + \left(\frac{3T_a T_{ATO}}{T_A} \times \frac{1}{T_{DIO}}\right) I_a V \tag{3.1}$$

where $V$ is the voltage of power supply, and $I_s$ and $I_a$ are the current consumptions during scanning and advertising, respectively. $T_{DIO}$ represents the DIO broadcast interval and $T_a$ is the packet transmission time. Here, the left term comes from the fact that a node scans an advertising channel for $T_W$ every $T_S$, which results in the default duty-cycle of $T_W/T_S$. The right term represents energy consumption for message transmission. An advertising node generates an advertising event $T_{ATO}/T_A$ times for a DIO message delivery. In addition, each advertising event has three packet transmissions over the three advertising channels, which results in the duty-cycle of $(3T_a \cdot T_{ATO})/(T_A \cdot T_{DIO})$ for DIO transmission.

Using the two conditions of $T_{ATO} = T_S$ and $T_A = T_W$, and letting

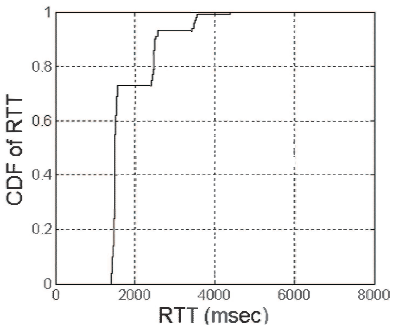
$X = T_W/T_S$, we can rewrite Eq. (1) as

$$P = I_s V X + \frac{3T_a I_a V/T_{DIO}}{X}. \tag{3.2}$$

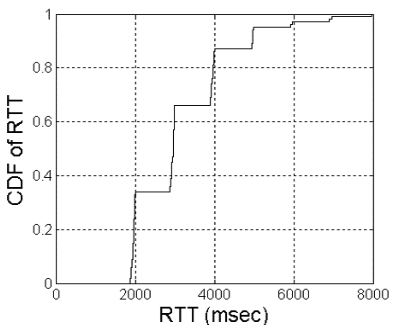
Then we can easily get an optimal value of $X$ that minimizes the power consumption as

$$X_{opt} = \sqrt{\frac{3T_a I_a}{I_s T_{DIO}}}. \tag{3.3}$$

Given that the current consumptions of a BLE node during receiving and transmitting modes are almost the same (i.e., $I_a \simeq I_s$) [2], we can further simplify $X_{opt}$ as $\sqrt{3T_a/T_{DIO}}$. Using this value, the LPL protocol can deliver DIO messages over advertising channels with minimal energy consumption.



(a) LOS link with 10 m distance



(b) NLOS link with 10 m distance

Figure 3.4: CDF of Logical Link layer ping's RTT with $T_{CI} = 1sec$.

### 3.3.3 Routing metric for RPL over BLE

Although RPL decouples routing metric from the standard specification, the most widely used routing metrics are end-to-end ETX and the combination of ETX and hop distance. However, these routing metrics cannot be applied to *RPL over BLE* since BLE link layer does not provide information about link layer retransmission for the upper layer. To help link quality estimation that is necessary to obtain routing metric, *ALBER* makes each child node periodically transmit a ping packet of Logical Link Control and Adaptation Protocol (L2CAP) to its parent node over data channels, and uses its round trip time (RTT).

Figure 3.4 shows the CDF of RTT values for link layer ping packets under two different link conditions (line-of-sight (LOS) and non-LOS (NLOS) links with the distance of 10 m) when the connection interval is 1 second. We use this figure to investigate feasibility of using an

RTT-based link quality metric and get an intuition about how to obtain a useful routing metric. We first observe that RTT values in the LOS link condition is much smaller than those in the NLOS link condition. This reveals that RTT reflects link quality indeed and has potential to be used as a link quality metric. Furthermore, we find out that RTT values are clearly quantized with the step size of connection interval, which comes from BLE's retransmission strategy. That is, when a BLE node fails to transmit a packet, it closes its current connection event and delays its retransmission attempt until the next connection event starts. This leads each retransmission to be delayed by one connection interval.

From these observations, we can simplify each measured RTT value as the number of connection intervals required for a packet transmission, i.e.,

$$N_{CI} = \left\lfloor \frac{RTT}{T_{CI}} \right\rfloor . \tag{3.4}$$

Then, we define a new metric, called ECI (Expected number of Connection Interval), as the exponentially weighted moving average of $N_{CI}$, and use it to represent link quality. Accordingly, node $n$ calculates its routing metric for its parent candidate $p_n$ as $R(p_n) = RANK(p_n) + ECI(n, p_n)$ where $RANK(p_n)$ is the end-to-end ECI between $p_n$ and the root, and $ECI(n, p_n)$ is the ECI between node $n$ and its parent $p_n$.

Our link estimation scheme has a trade-off relation between ping packet overhead and estimation accuracy. In this sense, we find out that the additional overhead is marginal compared to that of BLE's default connection management mechanism. Since a connected node sends a ping packet (21 bytes) instead of a null packet (10 bytes), each ping exchange requires only 176 $\mu$sec more radio-on time due to its 11 bytes larger packet size. In our testbed experiments, we empirically set the ping packet interval as 10 seconds, which results in only a 0.00176% increase in the duty cycle. Our performance evaluation shows that this ping packet interval is small enough to change routing topology according to link conditions, and our link estimation scheme is suitable to *RPL over BLE*.

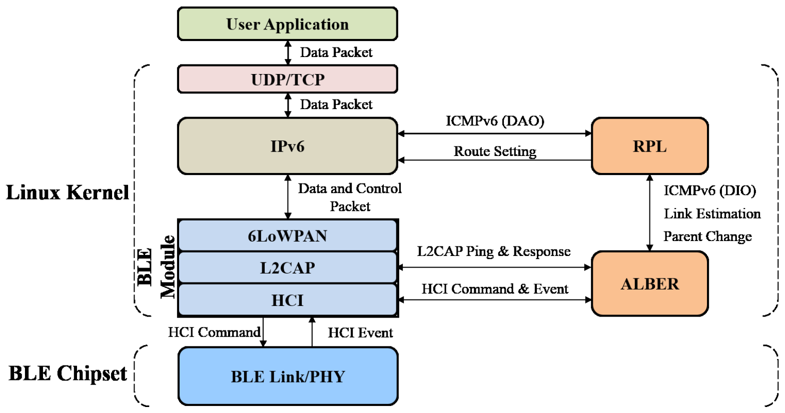


Figure 3.5: Protocol stack of *RPL over BLE* including *ALBER*.

### 3.3.4 RPL parent change with BLE connection management

Figure 3.5 shows the protocol stack of *RPL over BLE* that includes *ALBER*. Differently from *RPL over IEEE 802.15.4*, *RPL over BLE* allows a node to exchange IPv6 data packets through 6LoWPAN only with connected nodes. If the RPL of a node changes its parent node arbitrarily, 6LoWPAN starts to drop all the packets targeted at a new parent because it cannot find a valid connection with the new parent node.

To reduce inefficient packet losses, we design a seamless parent change mechanism for *RPL over BLE* as illustrated in Figure 3.6, which allows *ALBER* (rather than RPL) to determine whether to change a parent node considering BLE's connection status. Specifically, when RPL needs to select a new parent node, it notifies this event to *ALBER*, rather than updating the routing table hastily.

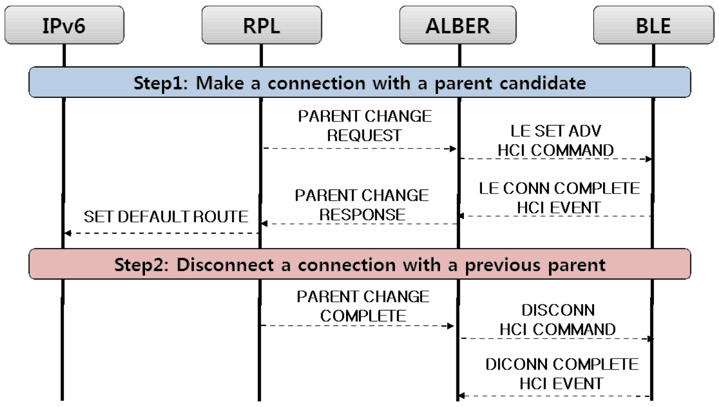

IPv6
RPL
ALBER
BLE
Step1: Make a connection with a parent candidate
PARENT CHANGE REQUEST
LE SET ADV HCI COMMAND
PARENT CHANGE RESPONSE
LE CONN COMPLETE HCI EVENT
SET DEFAULT ROUTE
Step2: Disconnect a connection with a previous parent
PARENT CHANGE COMPLETE
DISCONN HCI COMMAND
DICONN COMPLETE HCI EVENT

Figure 3.6: Connection-aware parent change procedures in *RPL over BLE*.

Then, *ALBER* makes BLE link layer establish a connection with the new parent node. After BLE link layer creates a connection, it reports *Connection Complete Event* to *ALBER*, which enables RPL to change the default route entry to the new parent node. When the BLE link layer fails to make a connection, RPL selects another parent node and repeats the same procedures again. Finally, after *ALBER* is reported that RPL finished updating the routing table, it makes BLE link layer cut off the connection with the old parent node and terminates the parent change procedures.

## 3.4 ALBER Implementation

We have implemented RPL over BLE in Linux kernel version 3.17 with the use of a BLE commercial chipset and the BLE module covering HCI, L2CAP and 6LoWPAN, which are available in the current kernel

Table 3.1: HCI commands for each operation of *RPL over BLE*

| **Operation** | **HCI commands** |
|---|---|
| RPL control message exchange | -LE Set Scan Parameter Command<br>-LE Set Scan Enable Command<br>-LE Set Advertising Parameter Command<br>-LE Set Advertising Data Command<br>-LE Set Advertise Enable Command |
| Parent change with connection management | -LE Create Connection Command<br>-LE Set Advertising Parameter Command<br>-LE Set Advertise Enable Command<br>-Disconnect Command |

version. We first implemented RPL in the Linux kernel based on ContikiRPL source code [40], and placed *ALBER* between RPL and the BLE module, as depicted in Figure 3.5. This hierarchical structure is designed for *ALBER* to perform required functions, and provides backward compatibility while minimizing modification of other layers.

Even though BLE's link and PHY layers (controller part) are generally implemented on a BLE hardware chipset, BLE provides various Host Controller Interface (HCI) commands, which helps the upper layer (host part) to manage some link level operations and determine lower layer parameters.

We design *ALBER* to use some of these HCI commands to interconnect RPL with BLE, as summarized in Table 3.1. Then, the HCI layer controls BLE's link layer operation when it receives HCI commands from *ALBER*. Moreover, it allows *ALBER* to have necessary link level information for the operation of *RPL over BLE* by reporting some link layer events such as connection establishment and advertis-

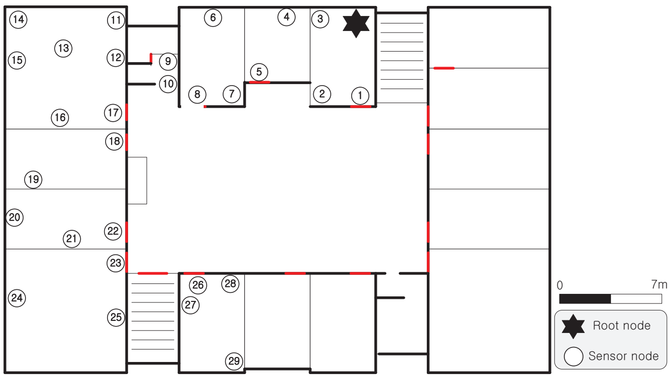


Figure 3.7: Testbed topology map with 29 nodes and one root.

ing packet reception. Finally, *ALBER* sends an L2CAP ping request to the L2CAP layer, and updates its ECI value by the received RTT value.

We modify ContikiRPL to support our *RPL over BLE* as follows. First, we make RPL send DIO messages through *ALBER* which achieves connection-less DIO message delivery over *advertising* channels[2]. Second, we design RPL to update its routing table after having *ALBER*'s approval. Finally, we use ECI (instead of ETX) as a routing metric to represent BLE's link quality.

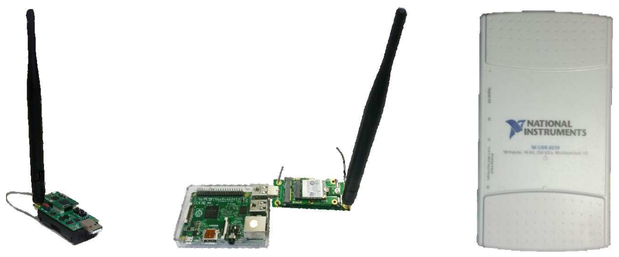


Figure 3.8: TelosB for IEEE 802.15.4 node (left), a set of Raspberry Pi and BCM4356 for BLE node (middle), and NI USB-6210 for power meter (right).

## 3.5 Performance Evaluation

### 3.5.1 Testbed environments

We configured a testbed topology in an indoor office environment as depicted in Figure 3.7, where a total of 31 nodes were deployed with one root node (marked with the star). For an IEEE 802.15.4 node, we use a TelosB clone device [41] with an MSP430 microcontroller and a CC2420 radio, and use a transmission power of 0 dBm with an antenna gain of 5 dB. For a BLE node, we use a Raspberry Pi device with Linux kernel version 3.17 and a Broadcom BCM4356 chipset as a BLE radio, with the same antenna gain of 5 dB as for an 802.15.4 node. Lastly, we use NI USB-6210 to measure the duty-cycle of a BLE radio since the BLE chipset does not provide this information. Figure 3.8 shows the devices we used in our experiments.

With the above hardware setup, we use a set of ContikiRPL and

[2]Note that DAOs and data packets in our design are delivered through IPv6 and 6LoWPAN.

ContikiMAC of Contiki 2.7 to evaluate the performance of *RPL over IEEE 802.15.4*, where the default sleep interval of ContikiMAC is 125 msec. For *RPL over BLE*, we use the connection interval of 50 msec (i.e., default value in HCI layer) and determine the parameters for scanning and advertising operations on *advertising* channels based on the optimization in Section 3.3.2. Although our mathematical analysis gives BLE's default duty-cycle ($T_W/T_S$) of only 0.1%, we practically exploit a sightly larger value of 0.2% in our experiments since the current BLE specification gives the parameter range requirements as $T_A(=T_W)$ $\geq$20 msec and $T_S$ $\leq$10.24 sec.

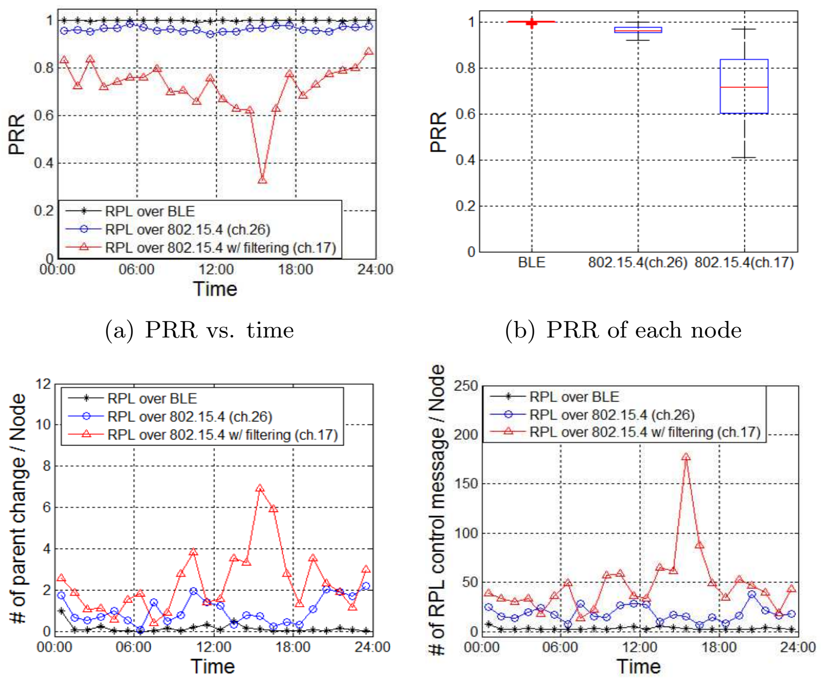
### 3.5.2 Comparison of RPL over BLE vs. RPL over 802.15.4

We first evaluate the effect of link dynamics on the performance of *RPL over BLE* by delivering a light upstream traffic[3] (5 minutes/packet/node) for 24 hours. For comparison, we also evaluate the performance of *RPL over IEEE 802.15.4* on both channels of 17 and 26[4].

Figs. 3.9(a) through 3.9(d) plot various performance metrics per hour. We first observed that *RPL over IEEE 802.15.4* on channel 17 performs too bad (i.e., packet reception ratio (PRR) of ~0%) during a day-time due to WiFi interference. Thus, we plot the results of another case that a node selects a parent node only when its received

[3]For brevity, we omitted the performance evaluation of downstream traffic delivery in this paper.

[4]Here channel 17 is with much heavier WiFi interference than channel 26.



(a) PRR vs. time (b) PRR of each node

(c) Parent changes vs. time (d) Routing overhead vs. time

Figure 3.9: Performance metrics of *RPL over BLE* and *RPL over IEEE 802.15.4* for 24 hours in a light traffic scenario (5 minutes/packet/node).

signal strength is higher than -70 dBm. Figure 3.9(a) shows that this filtering-based parent selection improves PRR performance on channel 17. However, it still shows the worst PRR all the time, and this PRR performance was further degraded during a day-time as low as ∼25%. This reveals that *RPL over IEEE 802.15.4* significantly suffers from WiFi interference.

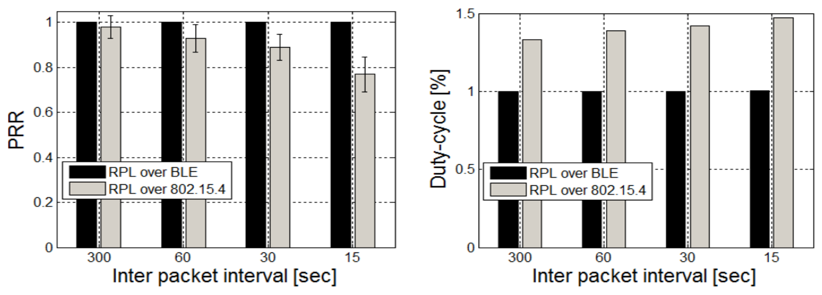
On channel 26, *RPL over IEEE 802.15.4* provides much better PRR. However, it still experiences some level of PRR degradation since ContikiMAC detects on-going packet transmissions using energy detection for an extremely short period[5], which incurs packet losses due to its low probability of energy detection.

On the other hand, *RPL over BLE* provides not only the best performance among competitors, but also almost perfect PRR performance all the time. This is because BLE's frequency hopping mechanism enables *RPL over BLE* to overcome link dynamics and WiFi interference. Moreover, owing to use of a synchronous MAC, a BLE node knows when its connected nodes wake up and send packets based on time synchronization, i.e., not energy detection. Lastly, Figure 3.9(b) reveals that *RPL over BLE* provides nearly perfect PRR fairness among the nodes while *RPL over IEEE 802.15.4* shows node location-dependent PRR due to each node's different link condition.

Figs. 3.9(c) and 3.9(d) show that *RPL over IEEE 802.15.4* takes a great effort to overcome link dynamics, which results in frequent parent changes and more routing control packet transmissions. However, the results in Figure 3.9(a) reveal that these efforts are not sufficient to reduce packet losses. In contrast, our *RPL over BLE* maintains quite stable routing topology and much lower routing overhead compared to *RPL over IEEE 802.15.4* all the time, which clearly shows a positive effect of BLE's connection-based operation.

In Figs. 3.10(a) and 3.10(b), we compare *RPL over BLE* and *RPL over IEEE 802.15.4* (on channel 26) under varying traffic load

[5]This design choice minimizes energy consumption for periodic wake-ups.



(a) PRR vs. traffic load

(b) Duty-cycle of a leaf node (node 30) vs. traffic load

Figure 3.10: Performance comparison of *RPL over BLE* vs. *RPL over IEEE 802.15.4* with varying traffic load.

in terms of PRR and duty-cycle[6]. In these experiments, we make the sleep interval of ContikiMAC equal to the connection interval of BLE (i.e., 50 msec) for fair comparison. As traffic load increases, *RPL over IEEE 802.15.4* experiences significant PRR degradation as low as ~80%. In ContikiMAC, a sender transmits packets repetitively during a whole sleep interval because it does not know when its receiver wakes up, which incurs contention and throughput degradation at heavy load even without WiFi interference.

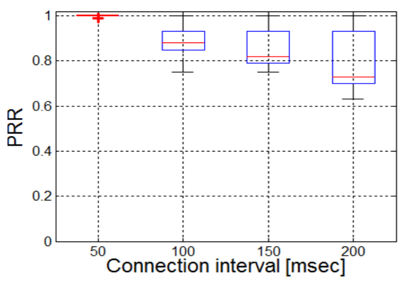
On the other hand, *RPL over BLE* maintains nearly perfect PRR regardless of traffic load since it does not require repetitive transmissions. Furthermore, in this architecture, we design a parent node (master) to schedule connection events of its children nodes, and to transmit control and data packets over different channel types, which

[6]We measured the duty-cycle of a leaf node (node 30) to focus on the effect of link layer operation on energy consumption, rather than on time-varying routing topology.

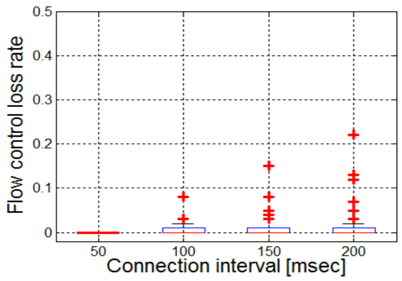
leads to contention-free data transmissions.

Figure 3.10(b) shows that *RPL over BLE* provides lower duty-cycle than *RPL over IEEE 802.15.4*, and the performance gain further increases up to 32%. This is because BLE's synchronous operation does not require repetitive (and redundant) transmissions, and its higher data rate (1 Mbps) helps to reduce packet transmission time. In addition, ContikiMAC has a false wake-up problem due to its energy-based packet detection and consumes more energy, which is not a problem in the case of BLE.

From the results of duty-cycle performance, we can infer three more things. First, given that the duty-cycle gain of *RPL over BLE* increases with traffic load, it benefits bottleneck nodes in a multi-hop network more significantly. Second, given that BLE consumes lower energy than IEEE 802.15.4 for transmission and reception, the energy saving effect of *RPL over BLE* overwhelms its duty-cycle improvement. Finally, given that the total duty-cycle of *RPL over BLE* was measured as 1% and the default duty-cycle ($T_W/T_S$) in advertising channels is 0.2%, we can infer that connection-based operation on data channels requires duty cycle of 0.8% (four times larger than the default one)[7]. This implies that we may further prolong battery lifetime by increasing connection interval.



(a) PRR of each node vs. connection interval



(b) Queue loss ratio of each node vs. connection interval

Figure 3.11: Performance of *RPL over BLE* with varying connection interval in a heavy traffic scenario (30 seconds/packet/node).

### 3.5.3 Effect of varying connection interval

Figs. 3.11(a) and 3.11(b) depict end-to-end PRR and packet loss ratio at link layer queue of each node, as the connection interval increases from the default value, when the data packet interval is 15 seconds/packet/node. Our results show that PRR decreases with connection interval since each node has less chances to transmit packets to its parent node. Specifically, a node with many children nodes can experience more queue losses since it receives too many packets during a connection interval without having any opportunity to transmit packets. Given that use of a larger connection interval has potential to save energy, we need to carefully determine the connection interval to achieve required throughput with minimal energy consumption.

Furthermore, from Figure 3.11(b), we observe that only a few nodes experience packet losses much more severely than other nodes.

[7]This includes periodic wake-ups and null (or data) packet exchange.

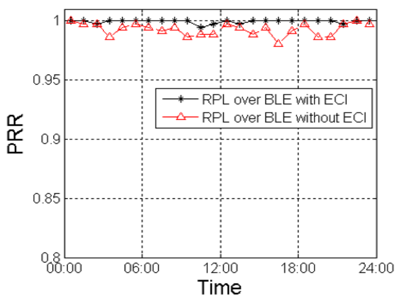
This unfair packet loss rate comes from the fact that a multi-hop network naturally makes nodes near the root node suffer much more relay burden and RPL has the load balancing problem [42]. This gives us an intuition that PRR performance of *RPL over BLE* can be improved by reducing the connection interval only for a few bottleneck nodes, without consuming large energy. From these results, we believe that connection interval control can be a valuable research topic for *RPL over BLE*.

### 3.5.4 Effect of ECI-based routing metric

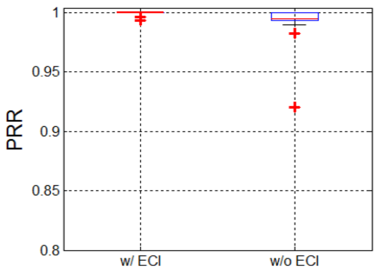
Now we investigate the effect of using ECI-based routing metric on the performance of *RPL over BLE*. To this end, we use Figs. 3.12(a) through 3.12(d) which plot various performance metrics per hour when *RPL over BLE* delivers light traffic (5 minutes/packet/node) for 24 hours with and without the proposed routing metric. Given that a BLE node does not feedback its link quality to the upper layers, we configure '*RPL over BLE* without ECI' to use only hop distance for its routing metric[8].

Figure 3.12(a) shows that our ECI-based routing metric for *RPL over BLE* leads to a slightly better average PRR all the time since it reflects link quality by using periodic ping packet transmissions. This can be verified in Figure 3.12(c), which shows that the proposed routing metric provides much lower link layer ECI by enabling each

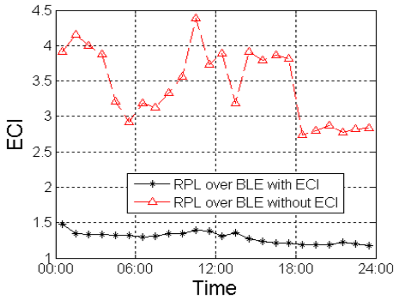
[8]This still measures ECI for performance comparison.



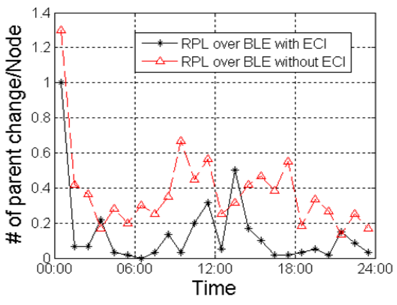
(a) PRR vs. time



(b) PRR of each node



(c) ECI vs. time



(d) Parent change vs. time

Figure 3.12: Performance of *RPL over BLE* with and without our ECI-based routing metric, for 24 hours in a light traffic scenario (5 minutes/packet/node).

node to select a parent node with good link quality. This enhanced parent selection stabilizes routing topology and results in a reduced number of parent changes as shown in Figure 3.12(d). Furthermore, Figure 3.12(b) reveals that the new routing metric improves PRR fairness among nodes. This confirms necessity of using the ECI-based routing metric by showing that, depending on physical topology, a certain node continuously suffers from packet losses without using link quality for routing metric.

## 3.6 Summary

In this paper, we interconnected RPL with BLE and constructed a BLE-based multi-hop network. Our approach was motivated by the potential that a standard-compliant multi-hop BLE network has great potential in the IoT domain. This is due to BLE being more accessible than IEEE 802.15.4, having a higher data rate compared to IEEE 802.15.4, using frequency hopping to mitigate interference, and synchronous operation.

We proposed a synergistic network architecture that incorporates a redesigned RPL control message broadcast, RPL parent change procedures, and RPL routing metric. This is captured by a new adaptation layer, termed *ALBER*, which enables tight coupling of RPL and BLE operations.

We evaluated the performance of *RPL over BLE* through extensive experiments in an indoor testbed. We compared the performance results with that of *RPL over IEEE 802.15.4* which showed significant improvement. We also showed that *RPL over BLE* is quite robust under varying link dynamics and WiFi interference. The proposed architecture significantly improves end-to-end packet delivery performance with low duty-cycle.

# Chapter 4

# Wi-BLE: A Novel Layered Architecture of Wi-Fi & BLE Networks for Disaster Communications

## 4.1 Introduction

The research on disaster communication network construction has been focused on the mobile wireless communication technology utilizing the base station infrastructure such as TETRA (Terrestrial Trunked Radio) and PS-LTE (Public Safety LTE) [43][44]. And the disaster network configuration focuses on communication support be-

Table 4.1: Comparison between Wi-Fi and BLE

| | **Wi-Fi (11n)** | **BLE** |
|---|---|---|
| Tx/Rx power consumption | 1224/302 mW | 95/95 mW |
| Tx power | 17.5 dBm | 10 dBm |
| Coverage | ∼100 m | ∼30 m |
| PHY data rate | ∼600 Mbps | 1 Mbps |

tween rescuers having dedicated terminals. However, when the existing communication infrastructure is destroyed by a disaster such as a large-scale earthquake, flood, or fire, ordinary people having only commercial smart phone device will fall into a state of incommunicable state.

In order to solve the communication isolation problem in the mobile communication shadow area, we proposed a multi-hop ad-hoc network configuration using a smart phone in disaster situation. For the smart phone based multi-hop network configuration, we can consider two candidate wireless technologies (i.e., Wi-Fi and BLE) and the characteristics of each communication interface are summarized in Table 4.1.

As shown in the table, Wi-Fi has wide transmission range and high data rate, while its high power consumption can significantly reduce the life time of smart phones in a disaster situation. On the other hands, BLE's low power consumption makes BLE be suitable for disaster communication where the lifetime of the smartphone is very important. However, BLE's low data rate is not enough to support the high data rate service like video call, which can limit the use of

BLE in disaster situations.

In this chapter, we have proposed a novel layered architecture of Wi-Fi and BLE (*Wi-BLE*) that uses both protocols together to meet the requirement of the ad-hoc network in a disaster situation. The main idea of *Wi-BLE* is that we can use BLE to construct the disaster multi-hop network with low energy consumption and turn on Wi-Fi interface only when the service requiring high data rate like video call is executed by a user. To do this, we first designed Mobile Ad-hoc for BLE (*MABLE*) layer to construct underlay BLE network with low power consumption. It uses routing protocol Ad hoc On demand Distance Vector (AODV) to find BLE route from a source to a destination node. Then, we can use low data rate service like text message exchange through this BLE path. When a high data rate service is executed, Wi-BLE layer triggers a Wi-Fi network configuration by delivering some control packet through the underlay BLE network. The Wi-BLE control packets make the nodes on the BLE path turn on their Wi-Fi interface and join the Wi-Fi ad-hoc network. Finally, *Wi-BLE* can support the high data rate service using the Wi-Fi network. The contribution of this chapter are four-fold:

- We propose a novel layered architecture of Wi-Fi and BLE networks that constructs an energy-efficient and high data rate supportable multi-hop network by using Wi-Fi on/off control that turn on Wi-Fi only when it is needed.

- We implement MABLE layer using routing protocol AODV to construct BLE multi-hop network which can support low data rate service and Wi-BLE control packet exchanges.

- For *Wi-BLE* operation, we define two types of Wi-BLE control packets and implement Wi-BLE layer over MABLE layer to deliver the control packet over the underlay BLE path. We also define two types of routing method of *Wi-BLE* and design a new path metric for the routing in Wi-Fi ad-hoc network.

- We experimentally investigate the performance of *Wi-BLE* compared to a conventional Wi-Fi network using AODV, which reveals that *Wi-BLE* can reduce the average power consumption of nodes by 80% without losing the high data rate support.

The rest of this chapter is organized as follows. In Section 4.2, we present our motivation to propose *Wi-BLE* and its use scenario in disaster situations. Section 4.3 describes the overview of *Wi-BLE* operation and its architecture. In Section 4.4, we implement MABLE layer to construct underlay BLE path. In Section 4.5, we define and proposed control packet format and routing method for *Wi-BLE* operation and Section 4.6 shows the detailed environment and work for *Wi-BLE* implementation. Then, we provide performance evaluation in Section 4.7, followed by summarizing remarks in Section 4.8.

## 4.2 Background

### 4.2.1 Application requirements of ad-hoc networks for disaster communications

This section summarizes the requirements to consider when configuring an ad-hoc network with smartphones for disaster communication in the event of a catastrophic collapse of a cellular network. First, low power consumption should be considered as an important requirement in order to remain communicable until the isolated people are rescued from disaster situations. Since the communication availability can be directly related to the survival rate of survivors, the energy consumed by the smartphone in order to maintain the ad-hoc network should be minimized. Next, providing a data rate above a certain level may be another requirement. In the event of a disaster, a text message exchange to inform the rescue signal becomes the main disaster communication service. However, in some cases, a service that requires a high data rate such as video transmission is needed in order to inform the rescue team about the appearance of the disaster site or the current state of the rescuer. However, the provision of high data rates and low energy consumption are basically conflicting concepts, so providing both together is a big challenge.

### 4.2.2 Candidate wireless interfaces for ad-hoc networks

When off-the-shelf smart phones are used to construct ad-hoc networks, we can consider two candidate wireless protocols (i.e. Wi-Fi and BLE). In the case of Wi-Fi, it provides high data rate which can satisfy the throughput requirement for Emergency video transmission in disaster situations. However, it's high power consumption accelerates the battery drain on smartphone, which can reduce people's survival rate. On the other hands, BLE's power consumption is under one-hundredth that of Wi-fi, which can dramatically increase the lifetime of smartphones. However, since the data rate of the BLE is only 1 Mbps which is not enough in some applications like video call, it also has a limitation to be used in a disaster network. Considering the pros and cons of each of these protocols, we proposed a layred architecture of *Wi-BLE* using both Wi-Fi and BLE to maintain low power consumption while providing high data rates in disaster networks.

### 4.2.3 Wi-BLE use scenario

When disaster destroyed existing cellular networks, people can use *Wi-BLE* to construct ad-hoc network using their own smart phones for disaster communication. Once users start to run *Wi-BLE* system on their smart phones, *Wi-BLE* first turn on only BLE interface to join an ad-hoc network using only a small amount of energy. After their smart phones succeed to build BLE ad-hoc networks, the users can use low data rate service like text message exchange using the

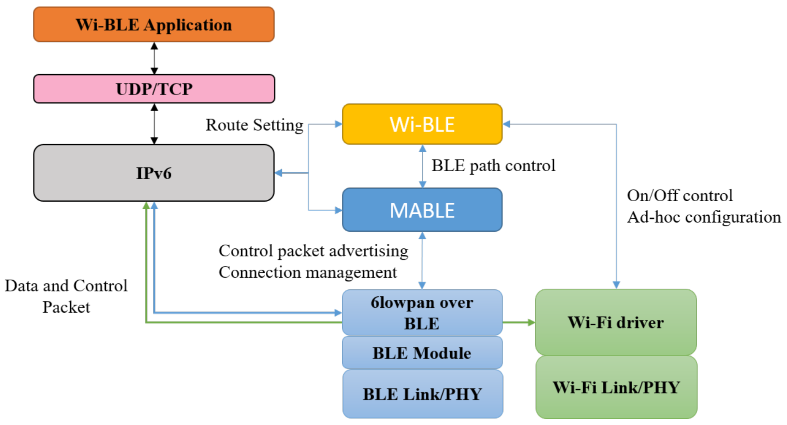


Figure 4.1: Protocol stack of *Wi-BLE*.

BLE-based multi-hop networks. Then, when the users want to execute a service that requires high data rate such as video transmission, *Wi-BLE* of source node generates a control packet which makes nodes in the path from source to destination to turn on the Wi-Fi interface. These node then construct a new Wi-Fi ad-hoc network based on the underlay BLE networks, which enables services requiring high data rates. In conclusion, *Wi-BLE* supports energy efficient network maintenance in disaster situations while providing high data rates as needed.

## 4.3 Wi-BLE System Overview

### 4.3.1 Protocol Architecture

Figure 4.1 shows the protocol stack of *Wi-BLE*. We have implemented *Wi-BLE* in Linux kernel and placed two layers (i.e., MABLE and Wi-

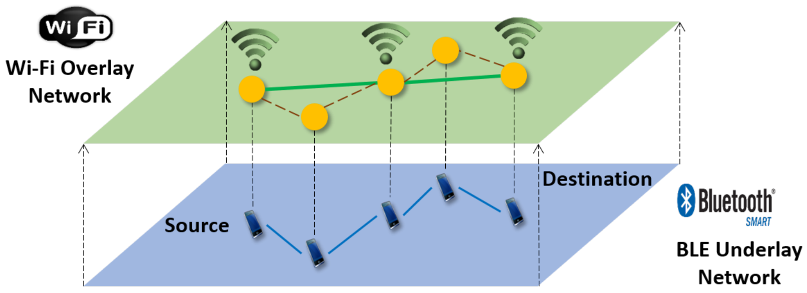


Figure 4.2: Operation overview of *Wi-BLE*.

BLE) that we designed for *Wi-BLE* operation. First, MABLE layer is placed between IPv6 and BLE module and responsible for creating and routing of BLE path. It cooperates with IPv6 layer for BLE route table setting and directly controls BLE module for advertisement of MABLE control packet and BLE connection management for BLE path creation. Wi-BLE layer must be enabled to turn on the Wi-Fi interface and configure the ad-hoc network according to the needs of the Wi-BLE application. To do this, it performs a Wi-Fi route table setting through the IPv6 layer and controls the operation of the Wi-Fi driver. In addition, it also involved in the BLE path generation of the lower layer MABLE to transmit the Wi-BLE control packets through BLE path.

### 4.3.2 Operation Overview

Figure 4.2 shows the operation overview of *Wi-BLE* with the layered network architecture of Wi-Fi and BLE. When Wi-BLE applications generates messages toward another node, they sends the packet to the IPv6 layer through the *Wi-BLE*'s BLE data port. Then, MABLE layer verifies the presence of a BLE path to the destination node and, if the path does not exist, initiates a task to create a new path. After successful path construction, Wi-BLE application can provide low data rate service such as emergency message exchange through the BLE path. If Wi-BLE application initiates a high data rate service like video transmission, it will send the corresponding data packets down to the Wi-Fi data port of *Wi-BLE*. Then, Wi-BLE layer creates a Wi-BLE control packet toward the destination node and the packet is delivered to the destination via the existing underlay BLE path of MABLE. This Wi-BLE control packet transmission causes all nodes in the path to turn on the Wi-Fi interface and configure the ad-hoc network, which enables the high data rate service of Wi-BLE application. Finally, *Wi-BLE* is able to satisfy both requirements (i.e., low energy consumption and high data rate) of disaster network since it usually maintains the networks with low power consumption using only BLE and turns on Wi-Fi interface only when the high data rate service is needed.

## 4.4 MABLE: Mobile Ad-hoc for BLE

### 4.4.1 Routing protocol selection for MABLE

In order to transmit text messages and configure a Wi-Fi overlay network based on an underlay BLE path, an appropriate routing technique is required to form a BLE multi-hop network. Such an ad-hoc routing scheme requires features such as low overhead and mobility support. Proactive routing schemes such as Destination-Sequenced Distance Vector Routing (DSDV) and Optimized Link State Routing (OLSR) are not suitable for disaster situations where energy constraints exist [45][46]. This is because in those routing schemes information of all nodes on the network must be periodically maintained, which increases the energy consumption of the network. In the case of the IPv6 Routing Protocol for Low-power Lossy Networks (RPL), which is the light-weighted table-driven routing schemes for wireless sensor networks, it is also inappropriate because it is reported that there is loop problems and performance degradation of reliability and delay on the networks where the mobile node exists. Also, because RPL assumes a DAG topology where traffic is directed to a single sink, it is not suitable for disaster communication scenarios where there are multi-flows with different sources and destinations. Among the reactive routing schemes, the DSR scheme is not suitable for BLE with limited frame length because it is based on source routing.

Another reactive routing scheme, AODV, does not have the problems of the above routing schemes, and its lightweight version, Lightweight On-demand Ad hoc Distance-vector Routing (LOADng), is being standardized in the IETF [47][48]. Therefore, we use AODV to construct BLE ad-hoc networks for *Wi-BLE*. AODV works in the following way. When traffic targeted to a destination with unknown routing path is generated, the source node floods a Route Request (RREQ) message to the entire network. The node which receives the RREQ records the node of the previous hop to the reverse route and rebroadcasts the corresponding RREQ message. If the same RREQ message is received, that message is not broadcasted again to prevent unnecessary flooding. When the RREQ reaches the destination, the destination node sends an RREP message to the source node to inform its existence along the reverse route formed by flooding of the RREQ. That is, through the unicast of the RREP, the nodes between the source and the destination learn the forward path.

### 4.4.2 BLE Channel Usage for AODV over BLE

#### Control packet delivery over advertising channels

BLE has two types of channel with different characteristics, so we have to choose which channel it is better to send routing messages such as RREQ or RREP. In advertising channels, the role of node is divided into advertiser and scanner. The Advertiser periodically broadcasts the same message, and the scanner receives the broadcasting message by periodically scanning three advertising channels. Since there is no synchronization between the advertiser and the scanner, the message is transmitted in a connectionless manner. In advertising channels, the length of payload is relatively short, up to 31 bytes, and there is no link layer acknowledgment, so it is mainly used for sending simple messages like beacon.

The data channel is based on a synchronized connection between the two nodes. When neighbor discovery is performed and a connection request is exchanged through the adverting channel, the two nodes are synchronized and share the periodic wake-up schedule and channel hopping sequence. A node that manages connection event scheduling and channel hopping sequence of the connection is called master node and the other node is called slave node. The data channel requires a synchronization overhead, but its payload length is up to 251 bytes, longer than that of the advertising channel, and it uses most of the channels (37 channels) used in BLE, which is advantageous in terms of throughput. In addition, the data channel has a link layer retransmission scheme and is robust to external interference through channel hopping, which is advantageous in terms of reliability.

In case of RREQ, it must be broadcasted to the whole network. Therefore, if RREQ message is sent through the data channel of BLE, all node pairs that can exchange RREQ message should make a connection at least once. This is unreasonable because it can complicate the wake-up scheduling of each master node and may make connections that do not participate in routing. In addition, the routing message is transmitted in the best effort manner via UDP and has a characteristic that its length is short. Therefore, *MABLE* exploits advertising channels to transmit routing messages and uses reliable and high-throughput data channels to transmit user traffic. *MABLE* transmits the routing message through the advertising chan-

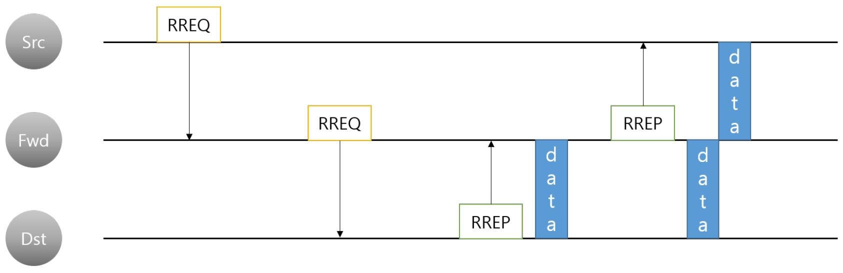


Figure 4.3: End-to-end connection establishment of BLE nodes.

nel, searches for the route path, and transmits data by establishing an end-to-end connection between the source and the destination.

**BLE path construction with a connection management**

To transmit user traffic from the source node to the destination node through the data channel, the path from the source to the destination have to be found and nodes on that path should establish BLE connections. These connections can be established with the reverse route setting when RREQ is flooded or with the forward route setting when the RREP is sent. If connections are constructed in reverse route setting, all nodes receiving RREQ can be potential forwarders, so all nodes need to make a connection with the previous node that relayed RREQ. This approach is unsuitable because as the network grows, the number of unnecessary connections that do not participate in the actual routing increases. On the other hand, in forward route setting, RREP is unicasted only between nodes participating in the actual routing path. Therefore, *MABLE* exploits forward route setting for connection establishment.

Figure 4.3 shows the procedure that *MABLE* constructs an end-to-end path between source and destination. The source node generates an RREQ message to search path to the destination and broadcasts it through the advertising channel. All the nodes on the network periodically scan the advertising channel. When an RREQ is scanned, it confirms that the destination is itself. If it is not the destination of the message, it stores the reverse route and re-broadcasts the message through the advertising channel. When the RREQ arrives at the destination through a series of RREQ broadcasting, the destination unicasts an RREP to the previous node that sent the RREQ to itself and prepares to establish a connection. The node receiving the RREP message establishes a connection with the node that sent the RREP, and records the forward route. Then, it relays the RREP to the next node on the reverse path generated by RREQ. By relaying this RREP in the direction of source, a hop-by-hop connection is established in turn along the reverse path. When the source node receives the RREP, an end-to-end BLE data channel connection is established.

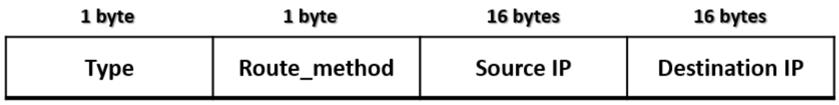


Figure 4.4: Wi-BLE control packet format.

## 4.5 Wi-BLE: Wi-Fi Ad-hoc over BLE networks

### 4.5.1 Wi-BLE control packet delivery over BLE path

When a Wi-BLE application executes a service that requires a high data rate, Wi-BLE layer generates control packet to request a configuration of Wi-Fi ad-hoc network over underlay BLE network. Then, it asks MABLE layer whether there is an available underlay BLE path connected to the destination of the service. If no path is available, *MABLE* starts to create a new BLE path and feedbacks the result to Wi-BLE layer. Finally, the source and destination node's Wi-BLE layer exchange the Wi-BLE control packet and try to build a new Wi-Fi ad-hoc network for high data rate service.

We define two types of Wi-BLE control packet used for *Wi-BLE* operation. First, when a user executes a Wi-BLE application that requires a high data rate, Wi-BLE layer generates Wi-BLE service REQuest (WREQ) packet. Then, the WREQ packet is delivered to the destination node through the BLE data path constructed by *MABLE*. When the destination node receives the WREQ, it create Wi-BLE service REsPonse (WREP) packet and forwards it to the source node through the BLE path. Then, nodes in the BLE path turn on their

Wi-Fi interface and construct Wi-Fi ad-hoc network.

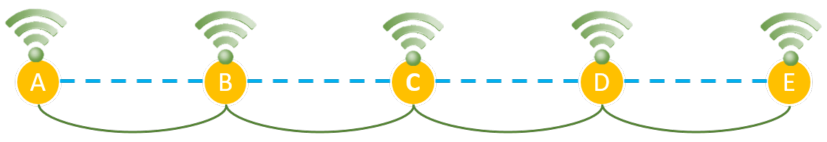
Figure 4.4 shows the packet format of Wi-BLE control packet. The type field contains the Wi-BLE control packet type (WREQ and WREP). When Wi-BLE layer receives Wi-BLE control packet through a BLE link, it checks the type field and starts packet processing. The second field shows routing method of Wi-Fi ad-hoc network and *Wi-BLE* defines two routing methods (USE_BLE_ROUTE and USE_WLAN_AODV). Details about the routing method is described in the next section. The last two fields contain the source and destination IP address of the newly created Wi-Fi path. Based on this information, Wi-BLE control packet can be routed through the underlay BLE path.

### 4.5.2 Routing protocol for Wi-BLE

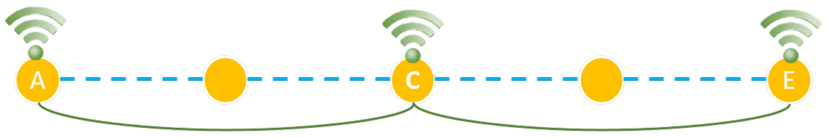
We define two routing methods (USE_BLE_ROUTE and USE_WLAN_AODV) for Wi-Fi ad-hoc network of *Wi-BLE*. In this section, detailed operation of *Wi-BLE* according to the routing method will be described.

#### Use BLE route for Wi-Fi ad-hoc networks

Figure 4.5 (a) shows an example routing topology of using USE_BLE_ROUTE routing method. In this figure, the blue dotted line and the green solid line indicate the BLE and Wi-Fi path, respectively. In the case of using USE_BLE_ROUTE routing method, Wi-Fi ad-hoc networks construct a new Wi-Fi path with the same routing path of underlay BLE



(a) Example of using USE_BLE_ROUTE method



(b) Example of using USE_WLAN_AODV method

Figure 4.5: Topology comparison of USE_BLE_ROUTE vs. USE_WLAN_AODV.

path. Thus, every nodes in the BLE path join a newly created Wi-Fi ad-hoc network and forward the data packets of a high data rate Wi-BLE application. When each node in the BLE path receives WREP packet with USE_BLE_ROUTE index from the destination node, it turns on Wi-Fi interface and sets the routing table of Wi-Fi path according to the existing BLE route. Therefore, the USE_BLE_ROUTE method does not requires a process to find a new routing path in a Wi-Fi ad-hoc network. On the flip side, it unnecessarily increases the number of hops in the Wi-Fi path due to the short coverage of BLE, which results in reduced end-to-end data rate and increased energy consumption of nodes.

### Use AODV routing protocol for Wi-Fi ad-hoc networks

When each node receives WREP with routing method USE_WLAN_AODV, it turns on Wi-Fi interface and starts to operate AODV routing protocol. Then, the source node floods the RREQ to the destination into the Wi-Fi network. After the destination node receives multiple RREQ packets over the various paths, it forwards the RREP to the source node through the path with the best path metric. Finally, the nodes forwarding the RREP set the routing path of the Wi-Fi network along the path of RREP. In this case, the node on the BLE path may not belong to the new Wi-Fi routing path created by AODV like in Figure 4.5 (b). Therefore, the USE_WLAN_AODV routing scheme can reduce the number of hops in Wi-Fi path, increasing the data rate of Wi-Fi path and reducing the energy consumption of the nodes.

In the case of using USE_WLAN_AODV, the path metric calculation and path selection of AODV greatly affects the performance of Wi-Fi network. Generally, the conventional AODV protocol mainly uses the number of hops of the path as a path metric. However, it includes a link with bad channel status in the path in order to reduce the number of hops, resulting very low data rate of the Wi-Fi path. Therefore, in *Wi-BLE*, we devise a path selection mechanism which considers not only the number of hops but also worst link's RSSI value. First, in order to prevent the link with poor channel condition being included in the path, the RSSI threshold is set so that the RREQ having a RSSI value lower than the threshold is dropped. Also, when RREQ is forwarded, the worst link's RSSI (denoted by RSSI_MIN) is put in the path metric field so that the destination node can refer to it when selecting the best path. Finally, the destination node selects the best path with minimum hop distance. However, if there are several paths with the same minimum hop distance, RREP is forwarded through the path having the highest RSSI_MIN among the candidate paths.

### 4.5.3 Wi-Fi on/off control for energy saving

*Wi-BLE* has Wi-Fi off timer to reduce energy consumed by unused Wi-Fi interfaces. For example, after a user finishes transmitting video over *Wi-BLE*, the terminals on the Wi-Fi path do not need to have their Wi-Fi interface turned on until the user starts the high data rate service again. Thus, if the Wi-Fi off timer confirms that there is no packet exchange through Wi-Fi interface for a predetermined time, it turns off the Wi-Fi interface and resets the routing table of Wi-Fi path. In our *Wi-BLE* implementation, we set the time limit of Wi-Fi off timer as 60 secs.

## 4.6 Implementation

We have implemented *Wi-BLE* in Linux kernel version 3.18 with the use of a commercial Wi-Fi and BLE chipsets. We first implemented MABLE layer in the Linux kernel based on ADOV source code, and placed MABLE between IPv6 layer and the BLE module part as depicted in Figure 4.1.Then, the Wi-BLE layer is implemented and placed over the MABLE layer, which makes *Wi-BLE* be able to control BLE path creation through the *MABLE*. Also, it is connected with IPv6 layer and Wi-Fi driver part in order to manage the operation and route table of Wi-Fi interface. This hierarchical structure is designed for *Wi-BLE* to perform required functions, while minimizing modification of other layers.

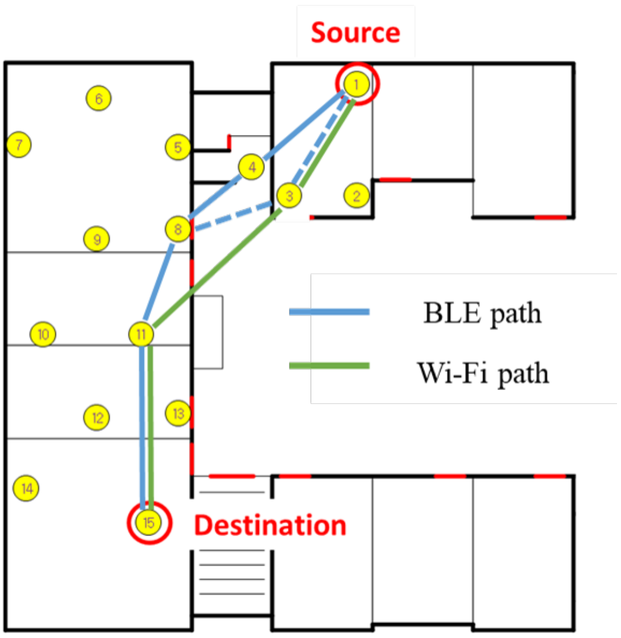
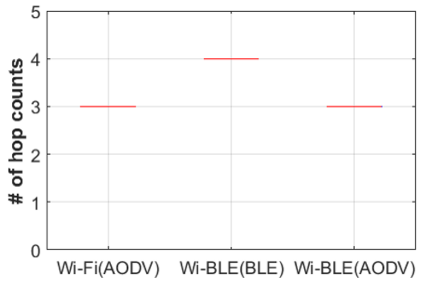


Figure 4.6: Testbed topology map with 10 nodes.

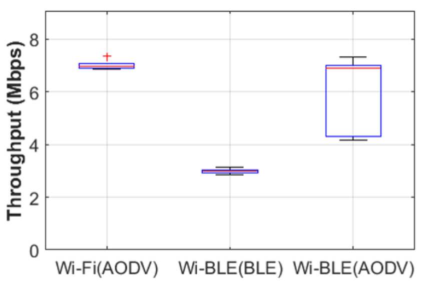
## 4.7 Performance Evaluation

### 4.7.1 Testbed Environments

We configured a testbed topology in an indoor office environment as depicted in Figure 4.6, where a total of 10 nodes were deployed with a source and a destination node. For each node, we use a Raspberry Pi2 device with a Broadcom BCM4356 and Atheros AR9271 for BLE and Wi-Fi chipsets respectively. For BLE link layer, we use the connection interval of 50msec (i.e., default value in BLE module of the Linux kernel) and Wi-Fi channel 11 with 20MHz bandwidth is used for Wi-Fi ad-hoc network configuration.



(a) Number of hops



(b) End-to-end throughput of Wi-Fi path

Figure 4.7: Performance of Wi-Fi ad-hoc network with three different schemes.

## 4.7.2 Hop distance & Throughput

We first evaluate the performance of *Wi-BLE* by measuring the hop distance and the throughput of Wi-Fi path from the source to the destination in our testbed. For comparison, we also evaluate the performance of AODV over Wi-Fi denoted by Wi-Fi(AODV) in Figure 4.7 where all the nodes in the network always turn on their Wi-Fi interface and find the routing path by using routing protocol AODV. In the case of *Wi-BLE*, there are two different routing method (USE_BLE_ROUTE and USE_WLAN_AODV) as described in 4.5.2. Thus, we compare the performance of *Wi-BLE* according to the routing method, which evaluates the effect of routing method. In Figure 4.7, *Wi-BLE* using each routing method are denoted by Wi-BLE(BLE) and Wi-BLE(AODV), respectively.

Figure 4.7(a) and (b) plot the number of hops and throughput of Wi-Fi path separately. First, the results regarding the hop distance show that Wi-Fi(AODV) and Wi-BLE(AODV) always have the same hop counts as three. On the other hands, the hop distance of Wi-BLE(BLE) has a higher value as four due to the short coverage of BLE. As a result, the Wi-Fi(AODV) and Wi-BLE(AODV) show the almost same throughput that is higher than the value of Wi-BLE(BLE). This is because the higher hop counts increases the contention in the Wi-Fi channel, which decreases the throughput of each link. However, we can see that the throughput of Wi-BLE(AODV) is occasionally lower than that of Wi-Fi(AODV). This is because Wi-BLE(AODV) can configure the Wi-Fi path only with the nodes in BLE path (blue lines in Figure 4.6), while Wi-Fi(AODV) can find the best Wi-Fi routing path with all the nodes in the networks (green line in Figure 4.6). Although the performance of *Wi-BLE* is less than that of Wi-Fi(AODV) in terms of throughput, we can see that this throughput loss is insignificant compared to the energy saving of *Wi-BLE* in the following power consumption results.

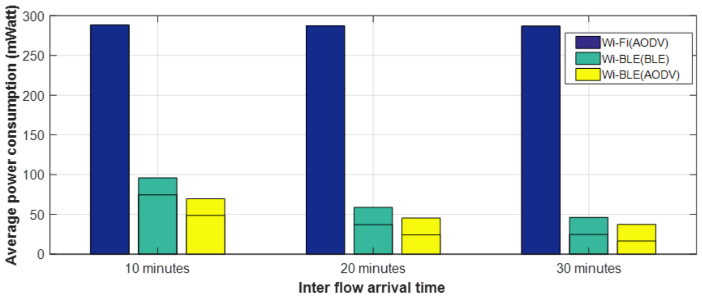


Figure 4.8: The average power consumption of nodes vs. flow arrival rate.

### 4.7.3 Power Consumption

To evaluate the energy consumption of *Wi-BLE*, we implemented a power monitoring thread in the Wi-BLE layer. The thread checks the operation state of the Wi-Fi and BLE interfaces at interval of 1msec and measures the power consumption of the communication interfaces based on the power consumption data of each operation state. In addition, it calculates the energy consumed by packet exchanges through each interface based on the length of the packets and the PHY data rates used for the packet exchanges. To do this, we simply modify some codes in the BLE module and Wi-Fi driver of the Linux kernel, which makes the feedback route for the packet exchange information from the lower layer to the Wi-BLE layer.

Figure 4.8 shows the average power consumption of nodes in the testbed according to the inter flow arrival time. To simulate intermittent high data rate service use in a disaster situation, we generated a 3Mbps UDP flow for each traffic interval using iperf application. The results show that Wi-Fi(AODV) consumes the highest power because it makes all the nodes on the testbed always turn on their Wi-Fi interface to join the ad-hoc network. In addition, we can see that the power consumption for the ad-hoc maintenance is significant, so the power consumption of Wi-Fi(AODV) does not change much with the traffic load. On the other hands, the power consumption of *Wi-BLE* linearly decreases according to the flow arrival time because it makes the nodes use the Wi-Fi interface only when they have to join the packet forwarding due to the high data rate flow occurrence. Finally, we can see that the *Wi-BLE* with USE_WLAN_AODV always shows the least power consumption even when compared to Wi-BLE(BLE). This is because it reduces the number of hops (i.e, the number of node participating in the packet forwarding) compared to USE_BLE_ROUTE case.

## 4.8 Summary

In this chapter, we proposed a novel layered architecture using Wi-Fi and BLE for disaster communication. Our approach was motivated by two requirements of disaster network (i.e., high throughput and low energy consumption). We designed *Wi-BLE* to maintain a disaster network with low power consumption and also provide high data rate occasionally.

For the *Wi-BLE* implementation, we designed *MABLE* first to maintain the underlay BLE network with low energy consumption. We also defined the format of Wi-BLE control packets (WREQ and WREP) and the two types of routing method to configure the Wi-Fi ad-hoc network as needed.

Finally, we evaluated the performance of *Wi-BLE* through extensive experiments in an indoor testbed. We compared the performance results with that of Wi-Fi network using AODV routing protocol and the results showed that our proposed *Wi-BLE* significantly reduces the energy consumption of nodes.

# Chapter 5

# Conclusion

## 5.1 Research Contributions

In this dissertation, we address two approaches to overcome BLE's range limitation and also suggest a new application scenario based on the extended coverage of BLE network.

First, we have addressed energy consumption and connection maintenance issues of BLE in dynamic wireless environments. We have considered the connection interval of BLE as the key parameter and control variable, focusing on how to effect adaptive control in dynamic channel environments. we have mathematically derived an optimal connection interval for a given channel condition and designed the *CABLE* system that estimates link quality and adjusts connection interval dynamically. We have evaluated the performance of *CABLE* both through extensive simulation and testbed experiments. The results show that significant performance improvement that closely approximates the optimum is achievable.

Secondly, we have constructed a BLE-based multi-hop network by interconnecting RPL with BLE. We have proposed a synergistic network architecture that incorporates a redesigned RPL control message broadcast, RPL parent change procedures, and RPL routing metric. We have evaluated the performance of *RPL over BLE* through extensive experiments in an indoor testbed. The results show that *RPL over BLE* is quite robust under varying link dynamics and WiFi interference and improves end-to-end packet delivery performance with low duty-cycle.

Thirdly, we have considered a new application scenario based on the multi-hop networking of BLE. We have proposed a novel layered architecture using Wi-Fi and BLE for disaster communication. We have designed *Wi-BLE* to maintain a disaster network with low power consumption and also provide high data rate occasionally. Finally, we have evaluated the performance of *Wi-BLE* through extensive experiments in an indoor testbed. We have compared the performance results with that of Wi-Fi network using AODV routing protocol and the results show that our proposed *Wi-BLE* significantly reduces the energy consumption of nodes in our testbed.

To summarize, the coverage extension of BLE has opened new possibilities to boost a penetration of BLE into the long range IoT applications. Although there still remain some issues to resolve, it is anticipated that BLE has significant potential for supporting longer range applications. Besides the issues in this dissertation, there remain many interesting problems, which require further investigation. This dissertation can be viewed as a guideline for overcoming the range limitation of BLE for future IoT networks.

## 5.2 Further Research Direction

Based on the results of this thesis, there are several new research directions which require further investigation. We highlight some of them as follows.

- **Time scheduling among connection events:** As the Bluetooth core 4.1 specification adopted multi-role capability, time resource management for multiple connections became a new issue for BLE. However, since the specification does not deal with any specific algorithm, BLE chipset companies have designed their own connection scheduling algorithms, which do not guarantee stable performance. For example, the BLE chipset of BCM4356 that we experimented often halted when more than five connections were involved simultaneously (e.g., one parent and four children nodes). To form a mesh that supports various IoT applications and deployment environments, a BLE chipset needs to support as many connections as possible, which calls for the development of a smart scheduling mechanism.

- **Mobility support in BLE mesh networks:** Even though our *RPL over BLE* provides reliable performance in relatively static scenarios, we cannot guarantee that it will work well for mobile applications such as group chatting and smart factory with mobile gateways (for instance, managers' smartphones). Due to the delay in parent change, the connection-based mesh solution may not be able to provide seamless multihop connectivity in dynamic environments. This issue needs to be further investigated under practical mobility scenarios. Moreover, the development of an effective connection-less mesh under mobility is an interesting topic.

- **Open source code and platform:** The protocol stack of BLE is divided into host and controller parts (i.e., upper and lower layers). The host part comprising upper layers (i.e., attribute protocol (ATT), security manger (SM), and logical link control and adaptation protocol (L2CAP)) is included in host devices. We can access and modify its source code for performance improvement. On the other hand, the controller part comprising lower layers (i.e., link and physical layers) is generally implemented on a commercial hardware chipset that is neither open nor modifiable. Different from IEEE 802.15.4 implemented on many open operating systems, BLE's poor accessibility has hindered its fast innovation and development. An open hardware platform for BLE was recently presented in [49], but given that it has implemented only the connection-less operation, its further development is imperative for the evolution of BLE.

# Bibliography

[1] Smart Home DB, http://www.smarthomedb.com/analytics

[2] S. Kamath and J. Lindh, "Measuring bluetooth low energy power consumption," Texas instruments, USA, Technical Report, 2010.

[3] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4," in *Proceedings of IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2012. pp. 232-237.

[4] T. Lee, M.-S. Lee, H.-S. Kim, and S. Bahk, "A synergistic architecture for rpl over ble," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, London, UK, Jun 2016.

[5] H.-S. Kim, H. Im, M.-S. Lee, J. Paek, and S. Bahk, "A measurement study of tcp over rpl in low-power and lossy networks,"

*Journal of Communications and Networks*, vol. 17, no. 6, pp. 647-655, 2015.

[6] J. Liu, C. Chen, and Y. Ma, "Modeling neighbor discovery in bluetooth low energy networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1439-1441, 2012.

[7] J. Liu, C. Chen, Y. Ma, and Y. Xu, "Energy analysis of device discovery for bluetooth low energy," in *Proceedings of IEEE Vehicular Technology Conference (VTC)*, 2013. pp. 1-5.

[8] K. Philipp, Y. Daniel, D. Robert, and C. Samarjit, "Precise energy modeling for the bluetooth low energy protocol," *arXiv preprint arXiv:1403.2919*, 2014.

[9] K. Philipp, Y. Daniel, G. Mathias, and C. Samarjit, "Adaptive Online Power-Management for Bluetooth Low Energy," in *Proceedings of IEEE INFOCOM*, 2015.

[10] C. Gomez, I. Demirkol, and J. Paradells, "Modeling the maximum throughput of bluetooth low energy in an error-prone link," *IEEE Communications Letters*, vol. 15, no. 11, pp. 1187-1189, 2011.

[11] CSRmesh, https://wiki.csr.com/wiki/CSRmesh

[12] P. Levis, T. Clausen, J. Hui, and O. Gnawali, The trickle algorithm, IETF, Technical Report RFC 6206, 2011.

[13] nRF51-ble-broadcast-mesh, https://github.com/NordicSemiconductor/nRF51-ble-bcast-mesh.

[14] K.-H. Chang, "Bluetooth: a viable solution for iot?[industry perspectives]," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 6-7, 2014.

[15] K. Mikhaylov and J. Tervonen, "Multihop data transfer service for bluetooth low energy," in *Proceedings of IEEE ITS Telecommunications (ITST),* 2013. pp. 319-324.

[16] B. K. Maharjan, U. Witkowski, and R. Zandian, "Tree network based on bluetooth 4.0 for wireless sensor network applications," in *Proceedings of IEEE Education and Research Conference (EDERC)*, 2014. pp. 172-176.

[17] Z. Guo, I. G. Harris, L.-f. Tsaur, and X. Chen, "An on-demand scatternet formation and multi-hop routing protocol for ble-based wireless sensor networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2015. pp. 1590-1595.

[18] Bluetooth 4.1 features & technical descriptions, http://www.bluetooth.com/

[19] Bluetooth core specification 4.2 frequentyly asked questions, http://www.bluetooth.com/.

[20] J. Decuir, "Introducing bluetooth smart: Part ii: Applications and updates." *IEEE Consumer Electronics Magazine*, vol. 3, no. 2, pp. 25-29, 2014.

[21] Using Bluetooth Products, http://www.bluetooth.com/Pages/using-bluetooth-products.aspx

[22] N. Gupta, *Inside BLUETOOTH LOW ENERGY*, ARTECH HOUSE, 2013.

[23] R. Heydon, *Bluetooth low energy: the developer's handbook*, Prentice Hall, 2012.

[24] Bluetooth Technology Makes Wireless Home Automation Possible,
http://www.bluetooth.com/Pages/Smart-Home-Market.aspx

[25] ALERT NOTIFICATION PROFILE,
https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx

[26] nRF Sniffer, https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy2/nRF-Sniffer

[27] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF, Technical Report RFC 6206, 2012.

[28] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks," in *Proceedings of the ACM SenSys Conference*, 2006.

[29] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Bounrdaries in Low-Power Networking," Stanford Information Networks Group, Technical Report SING-08-00, 2008.

[30] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proceedings of the ACM SenSys Conference*, New York, NY, USA, 2010, pp. 1-14.

[31] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol," SICS Report, Technical Report 5128, 2011.

[32] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of the ACM SenSys Conference*, 2009.

[33] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," in *Proceeding of ACM/IEEE Information Processing in Sensor Networks (IPSN)*, Apr. 2012. pp. 185-196.

[34] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in

*Proceeding of ACM/IEEE Information Processing in Sensor Networks (IPSN)*, 2011.

[35] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, “Low-Power Wireless Bus,” in *Proceedings of the ACM SenSys Conference*, Toronto, Canada, November 2012.

[36] Connected Grid Networks for Smart Grid - Field Area Network, http://www.cisco.com/web/strategy/energy/field_area_network.html

[37] K. Mikhaylov, N. Plevritakis, and J. Tervonen, “Performance analysis and comparison of bluetooth low energy with ieee 802.15. 4 and simpliciti,” *Journal of Sensor and Actuator Networks*, vol. 2, no. 3, pp. 589-613, 2013.

[38] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez, “ipv6 over bluetooth low energy,” IETF, Technical Report RFC 7668, 2015.

[39] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, ”Transmission of IPv6 packets over IEEE 802.15.4 Networks,” IETF, Technical Report RFC 4944, 2007.

[40] Contiki RPL code for Linux, https://github.com/joaopedrotaveira/linux-rpl

[41] Tmote Sky, http://www.moteiv.com/products/tmotesky.php

[42] H.-S. Kim, J. Paek, and S. Bahk, "QU-RPL: Queue Utilization based RPL for Load Balancing in Large Scale Industrial Applications," in *Proceedings of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Seoul, Korea, Jun 2015.

[43] P. Stavroulakis, *Terrestrial trunked radio-TETRA: a global security tool*, Springer Science & Business Media, 2007.

[44] T. Doumi, M. F. Dolan, S. Tatesh, A. Casati, G. Tsirtsis, K. Anchan, and D. Flore, "Lte for public safety networks," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 106-112, 2013.

[45] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *ACM SIGCOMM* , vol. 24, no. 4, pp. 234-244, 1994.

[46] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Technical Report, 2003.

[47] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," Technical Report, 2003.

[48] T. Clausen, A. C. de Verdiere, J. Yi, A. Niktash, Y. Igarashi, H. Satoh, U. Herberg, C. Lavenu, T. Lys, C. Perkins *et al.*, "The lightweight on-demand ad hoc distance-vector routing

protocol-next generation (loadng)," *draft-clausen-lln-loadng-09 (work in progress)*, 2013.

[49] S. Raza, P. Misra, Z. He, and T. Voigt, "Building the internet of things with bluetooth smart," *Ad Hoc Networks*, 2016.

# 초 록

저전력 블루투스 (Bluetooth Low Energy)는 사물인터넷 (IoT) 기기들 간의 무선 통신을 위한 대표적인 저전력 통신 프로토콜 중 하나이다. 저전력 블루투스 기술은 이미 시장에서 널리 사용되고 있으나 그것의 통신 범위 확장에 대한 연구 부족으로 현재는 통신범위가 짧은 통신 응용에서만 그 사용이 한정되어 있다. 본 학위 논문에서는 저전력 블루투스 네트워크의 통신 범위 확장에 관한 두 가지 연구 주제를 다룬다. 또한 최종적으로 저전력 블루투스 기반 멀티 홉 네트워크를 활용하기 위한 응용 시나리오를 새롭게 제안한다.

첫째, 통신 범위의 확장에 따라 기기간 거리가 멀어짐으로 인해 채널 환경의 변화가 심해지는 상황에서 저전력 블루투스 기기간의 연결 유지와 이때의 에너지 소모 최소화 문제를 다룬다. 그리고 이를 위해서 채널 상황에 맞게 저전력 블루투스의 링크 계층 파라미터 중 하나인 연결 주기($T_{CI}$)를 적응적으로 조절하기 위한 기법을 제안한다. 주어진 채널 환경에서 최적의 $T_{CI}$ 값을 찾기 위한 최적화 문제를 풀고 이를 기반으로 연결 주기의 적응적 조절 기법을 설계한다. 마지막으로 제안 기법의 성능 평가를 위해 테스트베드를 활용한 실험과 시뮬레이션을 진행한다. 그 결과는 제안하는 연결 주기 조절 기법이 채널 환경이 변하는 상황에서 저전력 블루투스 기기가 연결 유지를 위해 소모하는 에너지를 크게 줄임을 보여준다.

둘째, 저전력 블루투스의 통신 범위를 멀티 홉 네트워크 구성을 통해 더욱 확장시키기 위해 IPv6 routing protocol for low power and lossy networks (RPL)을 저전력 블루투스위에서 동작시키기 위한 프로토콜 구조를 제안한다. 라우팅 기법인 RPL과 저전력 블루투스 기술의 효율적인 연동을 위한 적응 계층을 새롭게 설계하고 실제 리눅스 커널 상에 구현한다. 성능 평가를 위해 실제 실내 테스트베드

상에서 다양한 실험을 진행하였으며, 제안하는 RPL over BLE의 성능을 비교 기법인 RPL over IEEE 802.15.4와 비교함으로써 제안기법의 뛰어난 성능 향상을 확인한다.
 마지막으로, 멀티 홉 네트워크 구축을 기반으로 한 저전력 블루투스의 통신 범위 확장을 활용하기 위해 새로운 응용 시나리오를 제안한다. 재난 상황에서 저전력으로 네트워크를 유지하면서도 필요에 따라 높은 데이터 수율을 제공할 수 있는 멀티 홉 네트워크 구축을 위해, 무선랜 (Wi-Fi)과 저전력 블루투스를 함께 활용하는 새로운 계층적 프로토콜 구조를 제안한다. 새롭게 제안된 프로토콜 구조를 실제 리눅스 상에 구현하고 실내 테스트 베드를 통해 그 성능을 확인한다. 결과를 통해 제안하는 기법이 기존 무선랜만을 사용하는 멀티 홉 네트워크에 비해 평균 에너지 소모를 크게 줄이는 것을 확인한다.

**주요어 : 저전력 블루투스, 사물인터넷, 저전력 블루투스 연결주기, 멀티 홉 네트워크, 재난 통신**
**학번 : 2011-20913**