The Dissertation Committee for Yi-Hung Wei
certifies that this is the approved version of the following dissertation:

Real-Time Communication Platform for Wireless Cyber-Physical Applications

Committee:

Aloysius K. Mok, Supervisor

Mohamed G. Gouda

Song Han

Simon S. Lam

Lili Qiu

Real-Time Communication Platform for Wireless Cyber-Physical Applications

by

Yi-Hung Wei, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2016

Dedicated to my grandfather, Tsao-Chi Wei (1933-2015), and my family.

Acknowledgments

First, I would like to express my greatest gratitude to my advisor, Prof. Aloysius K. Mok. Prof. Mok gave me great support and freedom to explore my interested topics, and provided insightful guidance whenever I need help. From him, I learn invaluable wisdom not only in research but also in life.

I would like to thank my dissertation committee members, Prof. Mohamed G. Gouda, Prof. Song Han, Prof. Simon S. Lam, and Prof. Lili Qiu for their precious suggestions and comments. I want to thank my collaborators, Prof. Masayoshi Tomizuka and Prof. Wenlong Zhang, for their tremendous support on building the wireless cyber-physical applications and providing valuable feedback on RT-WiFi communication platform. My appreciation also goes to all the members in my research group, Bing Ai, Wei-Ju Chen, Prof. Song Han, Yi-Hsuan Hsieh, Pei-Chi Huang, Quan Leng, Zheng Li, Jianyong Meng, Dr. Xiuming Zhu, and Lixun Zhang. Thank you for the fruitful discussions, and it is my pleasure to work with all of you. I especially like to thank Quan for collaborating with me on the RT-WiFi project. I will always remember the time we designed, developed, and debugged the RT-WiFi system together.

Thanks also goes to my friends for all the joyful memories in Austin: Chao-Yeh Chen, Chieh Chen, Yin-Yu Chen, Kai-Yang Chiang, Cho-Jui Hsieh,

Sze-Chi Huang, Wei-Lun Hung, Chia-I Lin, Yun Lin, Yumeng Ling, Yu Fong Wang, Si Si, Yu-Hao Tsai, Boris Yang, Hsiang-Fu Yu, and many others. I am grateful for your friendship and support.

Finally, I will never finish this dissertation without the endless love and support from my family. I would like to thank my grandfather, Tsao-Chi Wei, and my parents, Chu-Hsien Wei and Chin-Hsiu Hsu, my brother, Yi-Peng Wei, and my girl friend, Yi-Ting Wu. Thank you for your understanding, encouragement, support, and love. This dissertation is dedicated to all of you.

Yi-Hung Wei

The University of Texas at Austin
August 2016

Real-Time Communication Platform for Wireless Cyber-Physical Applications

Yi-Hung Wei, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Aloysius K. Mok

A Cyber-Physical System (CPS) is a physical system whose operations are monitored, coordinated, and controlled by computation and communication processes. Applying wireless technologies to cyber-physical systems can significantly enhance the system mobility and reduce the deployment and maintenance cost. Existing wireless technologies, however either cannot provide real-time or probabilistic guarantee on packet delivery or are not fast enough to support desired application requirements. Nondeterministic packet transmission and insufficiently high sampling rate will severely hurt application performance. To address this problem, we propose a real-time wireless communication platform called RT-WiFi. In this dissertation, we present our design and implementation of the data link layer and network management framework of RT-WiFi platform that provides predictable packet delivery and high sampling rate. The RT-WiFi communication platform is designed to support configurable components for adjusting design trade-offs including sampling

rate, latency variance, reliability and thus can serve as a suitable communication platform for supporting a wide range of wireless CPS applications. Based on the RT-WiFi management platform, we further propose advanced network management techniques to provide jitter-free scheduling algorithm for improving system performance and to support reliable data transmission in noisy environments. To evaluate the effectiveness of our proposed algorithms and to verify the efficiency of our network management platform, we conduct a series of experiments and a case study that integrate the RT-WiFi communication platform with a health care CPS application to investigate the application performance in the real world.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xiv**

List of Figures **xv**

Chapter 1. Introduction to RT-WiFi Wireless Communication Platform **1**

1.1 Introduction 1

1.2 Synopsis 5

Chapter 2. Related Work **8**

2.1 Wireless Protocols for Wireless Cyber-Physical Applications . 8

2.2 Data Link Layer Scheduling Algorithms 11

2.3 Reliable Communication Mechanisms 13

Chapter 3. RT-WiFi MAC Layer Design **16**

3.1 Introduction 16

3.2 RT-WiFi MAC Layer Protocol Design 19

3.2.1 Timer Design 22

3.2.2 Link Scheduler 24

3.2.3 Flexible Data Link Layer Design 26

3.2.3.1 Sampling rate 26
3.2.3.2 Reliability 27
3.2.3.3 Co-existence with regular Wi-Fi network 28
3.3 RT-WiFi MAC Layer System Implementation 31
3.3.1 Hardware and Software Platform 31
3.3.2 Timer Module 32
3.3.3 Link Scheduler and Message Handling Module 34
3.3.4 Flexible Channel Access Controller 35
3.4 Performance Evaluation 36
3.4.1 Performance Evaluation in Interference-free Environment 37
3.4.2 Performance Evaluation in Office Environment 40
3.4.3 Flexible Channel Access Controller 43
Chapter 4. Jitter-Free Scheduling in RT-WiFi Networks 47
4.1 Introduction 48
4.2 RT-WiFi Data Link Layer Communication Model 50
4.3 Static Link Schedule Assignment in RT-WiFi Networks 52
4.3.1 Jitter-Free Scheduling Problem 52
4.3.2 Harmonic Chain Based Jitter-Free (HCJF) Scheduler . . 54
4.3.2.1 Period Selection 55
4.3.2.2 Phasing Assignment 60
4.4 Dynamic Link Schedule Adjustment in RT-WiFi Networks . . 61
4.4.1 An Efficient Data Structure for Schedule Management . 63
4.4.2 *S*-tree Operations 66

4.4.2.1 Construct 66
4.4.2.2 Add-link 67
4.4.2.3 Remove-link 68
4.4.3 Schedule Management based on S-tree 68
4.4.3.1 Assignment policy 69
4.4.3.2 Replacement policy 69
4.4.3.3 Adjustment policy 70
4.5 Performance Evaluation 71
4.5.1 Simulation Model and Parameter Settings 71
4.5.2 Simulation Results 73
4.5.3 Network Utilization 75
4.5.4 Computational Overhead 77
Chapter 5. Enhancing Reliability in RT-WiFi Network 79
5.1 Introduction 80
5.2 System Model 81
5.3 Reliable Transmission for General Interference Source 85
5.3.1 Real-Time Reliable Link Scheduling Problem 85
5.3.2 Real-Time Rate Adaptation 88
5.3.3 Communication Link Scheduling 91
5.3.4 Schedulability Enhancement with Overbooking 94
5.3.4.1 Efficient Conflict Resolution Mechanism with Overbooking 95
5.3.4.2 Overbooking Scheduling 96
5.4 Reliable Transmission with Regular Wi-Fi Networks 99

5.4.1 Deferring Regular Wi-Fi Traffic with Virtual Carrier Sensing . . . 100
5.4.2 Sharing Unused Bandwidth with Regular Wi-Fi Networks 101
5.5 Performance Evaluation . . . 102
5.5.1 System Setup . . . 102
5.5.2 Rate Adaptation Algorithm Performance Evaluation . . 103
5.5.3 Performance Evaluation of Link Scheduler . . . 105
5.5.4 Co-existence with Regular Wi-Fi Networks . . . 108

Chapter 6. RT-WiFi Network System Implementation and Case Study 111

6.1 Introduction . . . 112
6.2 Design and Implementation of Network Management Platform 114
6.2.1 Network Management Architecture . . . 114
6.2.2 Network Manager . . . 115
6.2.3 Station Agent . . . 116
6.2.4 RT-WiFi Network Management Protocols . . . 118
6.3 Case Study - A Mobile Gait Rehabilitation System . . . 119
6.3.1 System Integration . . . 120
6.3.2 Emulation of a Wireless Control System . . . 122

Chapter 7. Conclusion and Future Work 127

7.1 Summary . . . 127
7.2 Future Research . . . 129
7.2.1 Multi-cluster RT-WiFi Network . . . 129

7.2.2 Failure Semantics for CPSs 129

Appendices **131**

Appendix A. Notational Conventions **132**

Appendix B. Acronyms **133**

Bibliography **134**

Vita **147**

List of Tables

3.1 Latency and packet loss ratio comparison in an interference-free environment 38
3.2 Latency and packet loss ratio comparison in an office environment 42
3.3 Latency and packet loss ratio comparison with present of large network traffic 43
3.4 Latency and packet loss ratio of RT-WiFi with different sampling rates in an office environment 45

5.1 Overbooking Conditions 96
5.2 Comparison of different co-existence mechanisms. 110

6.1 RMS tracking errors in simulations 126

List of Figures

1.1 Representative wireless protocols for networked cyber-physical systems . . . 2
1.2 Overview of the RT-WiFi communication platform design. . . 6

3.1 An example of an RT-WiFi-based wireless control system . . . 19
3.2 System architecture of RT-WiFi protocol . . . 20
3.3 Comparison of media access methods between regular Wi-Fi and RT-WiFi . . . 22
3.4 RT-WiFi time slot with successful transmission . . . 23
3.5 An example of a superframe with 8 time slots . . . 24
3.6 RT-WiFi time slot with in-slot retransmission . . . 28
3.7 RT-WiFi time slot when considering Wi-Fi traffic . . . 28
3.8 MAC layer transmission latency comparison between Wi-Fi and RT-WiFi in an interference-free environment . . . 39
3.9 MAC layer transmission latency comparison between Wi-Fi and RT-WiFi in an office environment . . . 41
3.10 Flexible channel access control testbed setting . . . 44

4.1 Tree representations of schedules. . . . 66
4.2 Overall Jitter . . . 74
4.3 Number of Adjustment . . . 75
4.4 Normalized Utilization . . . 76

5.1 An example of a communication link schedule. . . . 82
5.2 Overbooking example . . . 95
5.3 Co-existence mode . . . 99
5.4 Rate Adaptation Performance with Different Channel Condition. 103
5.5 Schedulability comparison of NP-EDF and SPF. . . . 105
5.6 Network management overhead Comparison. . . . 106

5.7 Schedulability improvement with overbooking technique. . . . 108
5.8 Communication schedules of co-existence experiment. 109

6.1 An overview of the RT-WiFi software architecture 115
6.2 An overview of the RT-WiFi management protocol. 117
6.3 An overview of the mobile gait rehabilitation system 121
6.4 Integration of smart shoes with a RT-WiFi station 122
6.5 ECDFs of tracking errors with $1Hz$ reference 124
6.6 ECDFs of tracking errors with $2Hz$ reference 125

Chapter 1

Introduction to RT-WiFi Wireless Communication Platform

1.1 Introduction

Cyber-physical systems (CPSs) that incorporate both physical components and computational components have drawn immense attention these years [66, 53]. In particular, for the computational subsystems to communicate with the physical world, wireless network systems play an important role to exchange critical monitor and control information for CPSs. Wireless network technologies have been adopted to numerous CPS because of their great advantages of enhanced mobility, easier deployment as well as reduced maintenance and configuration cost. For example, wireless technologies have been widely applied to industrial process control and automation [74], healthcare and biomedical systems [82], smart structures [55], and intelligent robotic systems [40], to name a few.

A key step in building a wireless CPS is to choose the most appropriate wireless communication protocol according to its desired application behavior. Application designers generally look at three most critical design factors when making the decision: sampling rate, reliability and real-time data delivery.

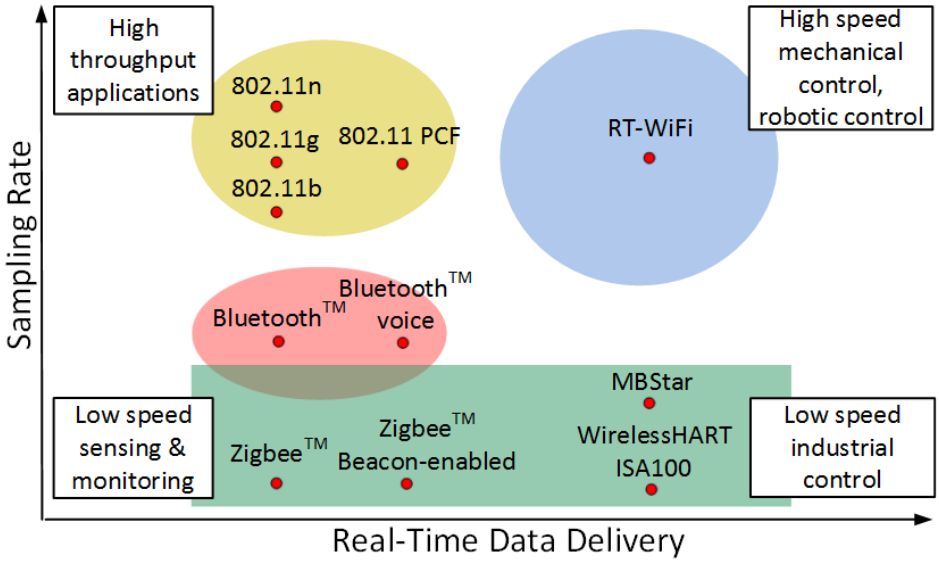


Figure 1.1: Representative wireless protocols for networked cyber-physical systems

These factors directly influence the quality of sensing and control data and the performance of the whole system.

Many existing wireless technologies have been adopted in wireless CPSs. However, each of these protocols can only support a narrow class of specific CPS applications as shown in Fig. 1.1. Some of them are based on low data rate physical layers, like IEEE 802.15.4 [4], and focus on the real-time data delivery and reliability performance; they are only suitable for low-speed control applications; others including Wi-Fi focus more on improving the network throughput but pay less attention to the deterministic behavior of data delivery, and are not suitable for CPSs that have stringent timing requirements. Taking our recent work [40] of designing a semi-autonomous remotely-

controlled robotic system as an example, since no available wireless technology can support both high data rate sensing flow and real-time control flow at the same time, we have to use a combination of two wireless technologies, Wi-Fi and WirelessHART [74], which complicates the system design, prolongs the development time and increases the maintenance cost.

To address the urgent need of a high-speed wireless protocol to provide predictable behavior for wireless CPS applications, in this dissertation we introduce RT-WiFi, a flexible real-time high-speed communication platform designed for supporting a wide range of CPSs. The fundamental building block of RT-WiFi communication platform is the RT-WiFi MAC layer[1]. The design goal of RT-WiFi MAC protocol is to support both predictable real-time data delivery with high sampling rates and application-customizable data link layer configuration. RT-WiFi networks adopt a centralized management mechanism which utilizes the time division multiple access (TDMA) channel access method for coordinating the channel access among RT-WiFi devices. By maintaining tight timing synchronization on each device, an RT-WiFi network divides the channel access time into small time slots, and the RT-WiFi network manager explicitly allocates channel access time slots for each device to achieve predictable data delivery. Since the physical layer of RT-WiFi provides sufficient bandwidth, RT-WiFi can support up to 6 kHz sampling rate. Furthermore, to meet diverse communication requirements of

[1]In the following of this dissertation, we use "data link layer" and "MAC layer" interchangeably.

various wireless control applications, we design RT-WiFi to be a configurable platform to provide great flexibility to a wide range of control applications.

Based on the RT-WiFi MAC layer design, we are developing more advanced network management techniques to support various wireless CPSs in different networking environments. While providing predictable latency is essential for wireless CPSs, the application performance is also highly impacted by communication jitter. To reduce jitter, we develop network management techniques to control network-wide scheduling of packet transportation. We propose efficient solutions for two fundamental RT-WiFi network management problems. To improve control performance in wireless CPSs, our RT-WiFi network manager is designed to generate data link layer communication schedule with minimum jitter under both static and dynamic network topologies. In order to minimize network management overhead, an efficient data structure is invented to manage the communication requests to deal with network dynamics.

Other than reducing communication jitter, we further consider how to provide reliable wireless communication in noisy environments. It is known that wireless communication is subject to uncontrolled packet loss due to noise and other interferences. In this dissertation, we explicitly consider interference from both Wi-Fi and non-Wi-Fi based interference sources, and propose two sets of effective solutions for RT-WiFi network management design. To improve reliability against general non-Wi-Fi based interference, we apply rate adaptation and retransmission techniques, and present an optimal real-time

rate adaption algorithm together with a communication link scheduler that has low network management overhead. A novel technique is introduced to further improve the schedulability of the communication link schedulers while maintaining the required communication reliability. For Wi-Fi based interference, we present mechanisms that utilize virtual carrier sensing to provide reliable data transmission while co-existing with regular Wi-Fi networks.

Finally, to evaluate the effectiveness of our proposed algorithms and to verify the efficiency of our system designs, we build the RT-WiFi management platform to dynamically manage and configure the associated RT-WiFi devices. The design of this management framework aims to meet the stringent real-time requirements of the RT-WiFi communication protocol, to be compatible with existing network configuration tools, and to provide clean modularization so that it can easily support different hardware and software configurations. Based on this communication platform, we conduct a series of experiments and a case study on a human rehabilitation system to demonstrate the benefit of our proposed system.

1.2 Synopsis

We present the structure overview of the RT-WiFi communication platform and the organization of this dissertation in this subsection. Fig. 1.2 depicts the structure overview of the RT-WiFi communication platform. The bottom of this figure shows two building blocks of the RT-WiFi infrastructure. We firstly build the RT-WiFi TDMA MAC layer to provide predictable

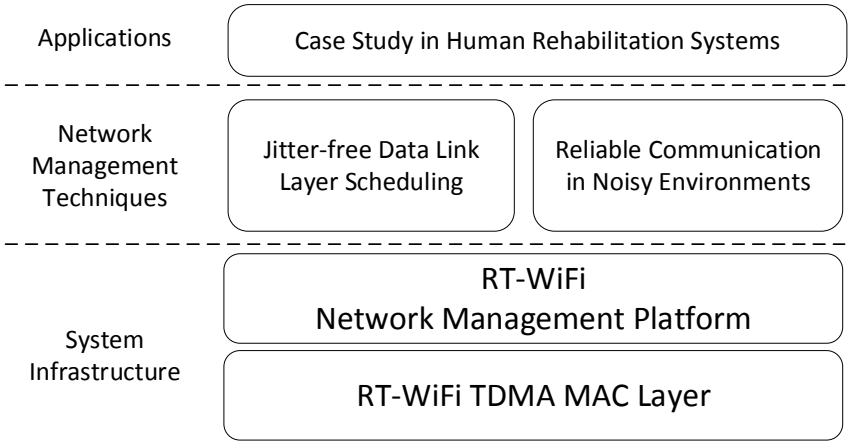


Figure 1.2: Overview of the RT-WiFi communication platform design.

real-time communication with high sampling rate. Based on the MAC layer design, we develop RT-WiFi network management platform that provides an interface for upper layer management software to dynamically configure the RT-WiFi system with specific requirements. On top of the RT-WiFi system infrastructure, we develop various network management techniques. We propose jitter-free scheduler to provide jitter-free data link layer communication schedule to improve application performance. To improve reliability against interference, we propose a set of reliable communication mechanisms for various interference sources. Finally, a case study on human rehabilitation application is built with RT-WiFi communication platform to evaluate the system performance.

The rest of this dissertation is organized as follows. In Chapter 2, we firstly compare and summarize existing wireless communication protocols, techniques for network resource scheduling, and reliable communication mech-

anisms. We present the RT-WiFi data link layer design in Chapter 3. In Chapter 4, we address the data link layer scheduling problem in both static and dynamic RT-WiFi networks. In Chapter 5, we consider the interference from both Wi-Fi and non-Wi-Fi interference sources, and develop various solutions to support real-time reliable communication for the RT-WiFi network. Chapter 6 reports the design and implementation of RT-WiFi network management platform, and presents a case study on health care CPS application. Finally, we conclude this dissertation and mark about avenues for future research in Section 7.

Chapter 2

Related Work

We review the related work in this chapter. We first summarize the existing wireless communication protocols that can be applied to wireless cyber-physical applications in Section 2.1. In Section 2.2, we present the available data link layer scheduling algorithms for real-time wireless resource management. Finally, in Section 2.3, we present the available wireless communication technologies that can be applied to provide reliable real-time data delivery.

2.1 Wireless Protocols for Wireless Cyber-Physical Applications

There are many existing wireless protocols which are designed for wireless cyber-physical applications. These protocols include Bluetooth [1] and ZigBee [13] for home automation, WirelessHART [74] and ISA100.11a [7] for industrial process applications, MBStar [83] for body area networks and so on. These protocols, however are not suitable for real-time high-sampling rate wireless cyber-physical applications. Bluetooth and ZigBee can provide real-time data delivery in particular modes, but due to their limited data rate they can only be applied for low-sampling rate applications. WirelessHART and

ISA100.11a are time division multiple access (TDMA)-based wireless protocols, which are specially designed for industrial process control applications. They can provide real-time communication, but the maximum supported sampling frequency is only $100Hz$. MBStar is designed to provide higher sampling rate than WirelessHART for body area networks. It can support up to 400Hz sampling rate, which is still much lower than the requirement in many high-sampling rate control applications.

ZigBee, WirelessHART, MBStar and the protocols proposed in [38, 67] cannot provide high enough sampling rate because their physical layer (IEEE 802.15.4 [4]) is designed for low power wireless personal area network. Comparing to the wireless communication protocol based on IEEE 802.15.4 [4], Wi-Fi, based on IEEE 802.11 [3], is a wireless protocol designed for high-speed wireless local area network and support data rates up to $1.6Gbps$ (IEEE 802.11ac with two spatial stream, 160MHz channel, and 256-QAM modulation type). However, regular Wi-Fi protocol does not provide any real-time guarantee on data delivery. Wi-Fi normally operates in the distributed coordination function (DCF) mode, in which each station applies a distributed randomized algorithm to access the channel for collision avoidance. Thus the delivery time of data is not deterministic.

There is another channel access method in Wi-Fi called point coordination function (PCF) [3]. In PCF, the Access Point (AP) operates as a point coordinator and determines which station can access the channel at a certain time. Unlike DCF, PCF is a centralized channel access method. The point

coordinator divides the time interval between two beacon frames into two periods: a contention-free period and a contention period. In the contention-free period, PC coordinates the data transmission among the stations and in the contention period, stations use DCF to access the channel. Although PCF can be used to coordinate packet transmission, it does not rely on any timing information, and thus cannot provide real-time data delivery guarantee. Also, once a station gets access to the channel, it may occupy the channel for a nondeterministic time interval. Thus, it is nondeterministic when the subsequent packets will be sent.

Providing real-time communication in Wi-Fi networks has attracted a lot of research interests in recent years. The HCF controlled channel access (HCCA) designed as a part of hybrid coordination function (HCF) is the medium access mechanism to provide hard QoS guarantees. But HCCA has a polling overhead and is shown to be inadequate for high-speed real-time control applications [21, 27]. In [61], an architecture based on virtual token passing procedure is proposed to enhance IEEE 802.11 with real-time packet delivery. It can co-exist with unconstrained devices and support stations join/leave the network. In [26, 28], a real-time communication architecture based on IEEE 802.11 is proposed and compared with HCCA. However, it only shows simulation results and does not provide the exact highest sampling rate that can be supported for wireless CPS applications.

Time division multiple access (TDMA) is a channel access method which relies on accurate timing information among the communication de-

vices and may potentially provide real-time data delivery. Some TDMA-based protocols [50, 31, 68, 65, 29, 75] have also been proposed based on IEEE 802.11. Both [50] and [31, 32] focus on the time synchronization issue in multi-hop networks. [68, 65, 29] enhance IEEE 802.11 MAC to support bandwidth-efficient long-range wireless communication in rural areas. They do not provide a flexible platform for control applications, and do not study the co-existence issue with regular Wi-Fi networks. [75] presents a Wi-Fi-based TDMA implementation, and investigates different timer sources for TDMA implementation in the Linux system. [75] provides valuable implementation details, but it does provide the synchronization mechanism between AP and stations and the TDMA network management structure. [69] implements an overlay MAC layer to address some of the limitation of IEEE 802.11 on throughput, fairness, and interference. [15] increases the capacity of IEEE 802.11 ad-hoc network by exploiting frequency diversity. Both [69] and [15] are not designed for real-time applications and thus cannot provide deterministic real-time data delivery.

2.2 Data Link Layer Scheduling Algorithms

To guarantee the real-time data delivery in RT-WiFi networks, the network management techniques should consider both the data link layer schedule assignment and its adjustment to network dynamics. Several network management schemes have been proposed in TDMA-based real-time wireless networks, but cannot be directly applied in RT-WiFi networks. For example, [41] focuses on constructing reliable routing graphs for different communication purposes

in industrial wireless mesh network, while [70] studies how to assign priorities to flows in real-time wireless networks and analyze their end-to-end delay. [16] assumes the transmission time of a communication task is one, and provides solutions to select task periods and a tree structure to efficiently store the communication schedule. [33] presents a time slot allocation algorithm to minimize jitter for TDMA network, but this work does not consider flexible task period and dynamic schedule adjustment. [64] proposes an efficient delay-bounded scheduling based on angular interference model for TDMA long-distance WiFi network. [20] presents a contention-free TDMA scheduler for scalable wireless sensor network. However, all aforementioned works consider different perspectives of the network management problem, but did not address dynamic network management and the special requirements of control applications.

There are several processor scheduling algorithms which may be applied to the real-time high-speed data link layer scheduling problem. For example, [59, 17, 57] focused on minimizing jitters for static task set. [73, 84] consider the tasks with a range of acceptable periods, and propose solutions for adjusting task periods to meet the control application performance or the latency requirements. [58] provides schedulability analysis for non-preemptive jitter-free processor scheduling, however, it cannot be directly applied to schedule tasks with flexible periods, and dynamic network environments. [51] discusses the load adjustment of periodic tasks, and proposes a utilization bound that is a function of the number of harmonic chains in the task periods. [23] improves the utilization bound by considering extreme and critical task sets separately

and by utilizing the task parameters. [18] proposes P-TIME schedulability tests for sporadic task sets with harmonic periods. [22] further considers to adjust both the task periods and task deadlines at the same time, and proposes selection heuristics. All aforementioned research work, although consider different types of tasks adjustment, either do not consider dynamic network management overheads or do not address how to eliminate or minimize the jitters, which is a key performance metric in high sampling rate networked control systems.

2.3 Reliable Communication Mechanisms

There have been several works that aim to provide reliable data transmission in regular Wi-Fi networks. For example, rate adaptation is an effective technique that adjusts the data rate of a link and employs a more error resilient coding scheme. Several works [81, 46, 72] have been proposed to utilize rate adaptation to enhance reliability in wireless networks. However, the aforementioned works are focusing on improving the overall network throughput, and do not differentiate the application-specific real-time constraint on each flow. By considering rate control, frame aggregation, and retransmission dispatching, [56] focuses on reduce the long tail of the packet delay distribution, but it does not consider the real-time characteristics of each communication link as well. On the other hand, [43, 44] present a framework for flow scheduling, and their work may be applied to enhance quality of service in Wi-Fi network. However, their focus is on achieving long-term fairness for network flows, and

do not focus on enhancing real-time, short-horizon transmission reliability.

Several works[71, 25, 41, 24] have been proposed to provide reliable transmission in both IEEE 802.11 based and wireless sensor network based TDMA networks. [71] proposes an IEEE 802.11 based TDMA protocol that reduces data collisions and adapts its transmission schedule to channel conditions. [25] presents an admission control mechanism for IEEE 802.11 networks that considers the transmission errors and medium access delays caused by uncontrolled traffic. [41] considers the reliable transmission in multi-hop mesh sensor networks, and discusses how to construct a reliable routing graph with duplication in communication paths. [24] considers the interference graph when in constructing multi-hop schedulers. However, these past works focus on optimizing a particular reliability requirement, they either do not consider the rate adaptation mechanism or interference from regular Wi-Fi networks.

If the transmission time of each task is pre-determined, then scheduling communication links is similar to scheduling non-preemptive tasks on a CPU. There are several works in the non-preemptive scheduling literature that can be applied to the reliable data link scheduling problem. [45] shows that the problem is NP-hard in the strong sense to schedule a set of non-preemptive concrete task sets on an uni-processor. If the tasks are not strictly periodic, [19] shows that the non-preemptive scheduling problem remains NP-hard even if the task periods are harmonic. Using the knowledge of incoming task, [36] proposes a scheduler to insert idle time for increasing schedulability. Both [19] and [63] present optimal schedulers with certain strong restrictions on the task

periods. However, the periods of communication links usually may not fit into their assumption. If the tasks are strictly periodic, [58] provides sufficient conditions for scheduling task sets with both harmonic or non-harmonic periods. [49, 35] presents algorithms to determine the minimal number of processors to schedule different task sets. All the previous works provide great insights into understanding the difficulty of the link scheduling problem.

Although the CPU scheduling problem is well studied, the network link scheduling problem is different from CPU task scheduling problem significantly in terms of scheduling overhead. While the scheduling overhead of CPU scheduling is relatively small, scheduling network links involves coordination among multiple entities. To schedule a network link, the link scheduler sends multiple network management messages to make all the nodes in the network to reach consensus on channel usage. Therefore, in addition to studying schedulability on the link scheduling problem, this dissertation also focuses on minimizing the network management overhead in order to distribute communication schedules efficiently.

Chapter 3

RT-WiFi MAC Layer Design

To build a high-speed real-time communication platform for wireless cyber-physical applications, in this section, we introduce the key building block of RT-WiFi communication platform, the RT-WiFi MAC layer design. We first introduce the background information in Section 3.1. We then present RT-WiFi MAC layer design in Section 3.2. We report our implementation based on real hardware architecture in Section 3.3, and summarize the performance evaluation in Section 3.4.

3.1 Introduction

The design goal of RT-WiFi MAC layer is to provide both real-time high-sampling-rate data transmission and application-dependent flexible data link layer configuration. Real-time data delivery is crucial in CPSs because applications rely on precise timing for monitoring and controlling the state of systems. High sampling rate is required to achieve good quality of control and to support ever-increasing number of sensors and actuators in a control

This chapter is previously published in [77]. I contributed MAC layer design, system implementation to this published work.

system. For supporting a wide range of control applications, the design of RT-WiFi MAC layer shall consider various requirements for different types of control applications. The design philosophy of RT-WiFi MAC layer is to provide enough freedom for control designers to choose their preferred communication behavior that best fits their controller design. At the same time, the design of RT-WiFi MAC layer protocol should minimize the modification on the original Wi-Fi protocol, so that it can be transparent to both the upper layer software stack and underlying hardware, and thus provide the most compatibility and usability. To address the need of a high-sampling-rate wireless protocol to provide predictable behavior for cyber-physical applications, we keep the following goals in mind when we design the RT-WiFi MAC layer:

- **Real-time Data Delivery and High Sampling Rate**: Control applications rely on bounded latency for estimating and controlling the state of the target system. Our previous research [78] shows that unlike relatively low-speed process control applications, mechanical control systems usually require a sampling rate higher than $1kHz$. To achieve deterministic timing behavior, we adopt time division multiple access (TDMA) mechanism in the RT-WiFi data link layer design for channel access and set stringent timing requirements on the transactions in each time slot. To support high sampling rate, RT-WiFi MAC layer is built on top of IEEE 802.11 [3] physical layer.

- **Flexible Data Link Layer Configuration**: While different control

applications have different communication requirements on data delivery, it is hard to apply a rigid design to a wide range of applications. We envision RT-WiFi as a configurable platform to provide great flexibility to the application designers, who can choose their own design parameters according to the target applications. These design trade-offs include sampling rate, communication reliability, real-time data delivery, and co-existence with existing Wi-Fi networks.

- **Transparent System Design**: In order to make RT-WiFi easily available and simple to integrate with existing applications, it should reuse current hardware and make minimum modifications on existing software. Users should be able to use RT-WiFi MAC layer protocol on commercial off-the-shelf (COTS) IEEE 802.11 network interfaces, and run existing applications on top of RT-WiFi with no or minimum changes. In addition, the transparent system design allows the users to leverage existing IEEE 802.11 security techniques and build mesh networks easily.

Figure 3.1 shows the typical architecture of a control system that adopts RT-WiFi as the communication infrastructure. In this example, a system architect plans to deploy three sensors and three actuators in the system. Depending on the physical proximity of each sensor/actuator, they are attached to different RT-WiFi stations. In an RT-WiFi network, a network manager and control applications are running on top of the RT-WiFi AP. The network manager is responsible for configuring the RT-WiFi network based on

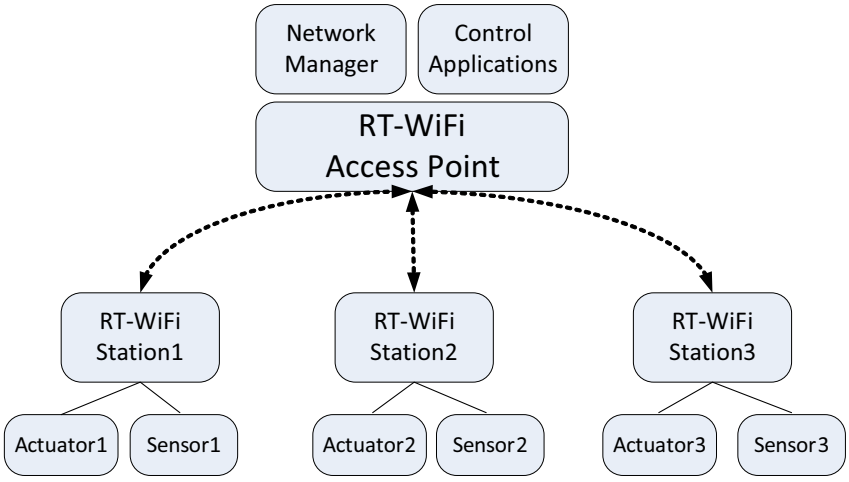


Figure 3.1: An example of an RT-WiFi-based wireless control system

the communication requirements specified by a control designer. The network manager dynamically allocates communication resource and configures the data link layer of RT-WiFi AP and stations properly. After the configuration stage, the RT-WiFi network will enter the operational mode. Sensor data and control data will be exchanged periodically between the RT-WiFi stations and the control applications. In the following of this section, we will focus on the RT-WiFi data link layer design and implementation.

3.2 RT-WiFi MAC Layer Protocol Design

In this section, we present the design details of the RT-WiFi MAC layer. RT-WiFi data link layer protocol is based on IEEE 802.11 physical layer and aims to provide real-time data delivery for a wide range of wireless control systems from low-speed industrial process control to high-speed mechanical

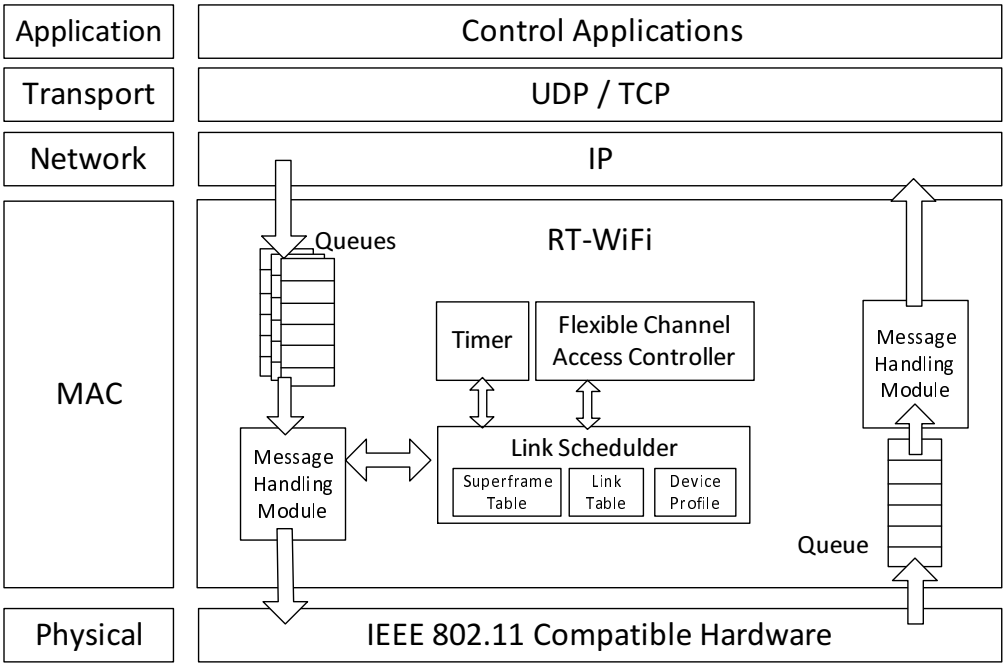


Figure 3.2: System architecture of RT-WiFi protocol

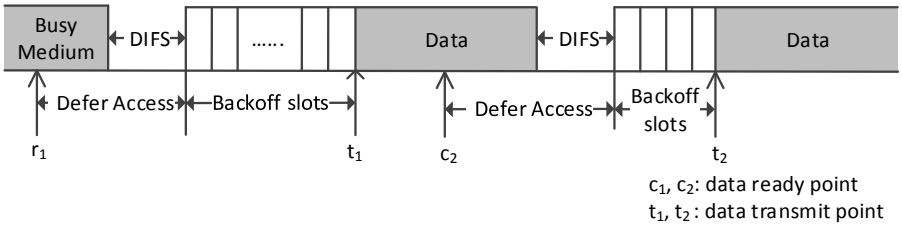
device control.

Fig. 3.2 presents the overall architecture of RT-WiFi MAC layer protocol. At the very bottom, RT-WiFi adopts the physical layer of IEEE 802.11 [3]. IEEE 802.11 is one of the most pervasively adopted wireless technologies, and it supports sufficient bandwidth for most wireless control systems. Control application users can easily deploy the RT-WiFi data link layer on a commercial off-the-shelf IEEE 802.11 compatible hardware for supporting high-sampling-rate and real-time data delivery. Sitting on top of the IEEE 802.11 physical layer is the RT-WiFi data link layer which we shall elaborate in the rest of this section. The RT-WiFi data link layer provides a flexible abstraction for the

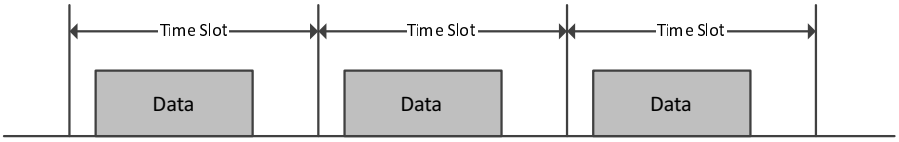
upper layers; thus, standard UDP or TCP-based applications are supported seamlessly. Various control protocols can be readily supported by wrapping their control payload in TCP/UDP packets.

The core of the RT-WiFi MAC protocol is the TDMA-based design. Regular Wi-Fi network adopts carrier sense multiple access with collision avoidance (CSMA/CA) to avoid collision. This mechanism helps improve the network throughput but is not feasible for supporting predictable real-time traffic with stringent timing requirements. Fig. 3.3 provides a comparison of the media access methods between regular Wi-Fi and RT-WiFi. In Fig. 3.3a, the transmission of the regular Wi-Fi packet with a hard deadline may be blocked for a nondeterministic time interval because of carrier sense. Packet transmission may be further delayed by the random backoff mechanism. For example, in Fig. 3.3a, data transmissions that are available at time r_1 and r_2 are deferred to transmit at time t_1 and t_2 in regular Wi-Fi network respectively. To address this problem, we adopt TDMA mechanisms and centralized channel and time management schemes to access the channels according to a strict time schedule. Fig. 3.3b presents the channel access pattern of a RT-WiFi network. Since only one node can access a certain channel in a given time slot, RT-WiFi provides collision free and deterministic communication.

There are three key components in the RT-WiFi data link layer: a *timer* for maintaining global synchronization among all RT-WiFi nodes and triggering timing events; a *link scheduler* for coordinating the access to shared media, and executing the dedicated event at scheduled time points; and a



(a) Regular Wi-Fi



(b) RT-WiFi

Figure 3.3: Comparison of media access methods between regular Wi-Fi and RT-WiFi

flexible channel access controller which dynamically configures the hardware parameters for executing the timing event according to the behavior of target application. We shall explain the design of each component in the following sections.

3.2.1 Timer Design

To achieve high sampling rate, RT-WiFi has stringent timing requirements on each device. For instance, if the required aggregate sampling rate is $5kHz$, the timer is required to deliver an accurate timer interrupt for every $200\mu s$, and all tasks related to that sample (including data transmission, acknowledgement and possible in-slot retransmission) shall be completed within that short interval (called a *time slot*). A time slot is a basic temporal channel access unit in the RT-WiFi network. To provide deterministic timing behavior

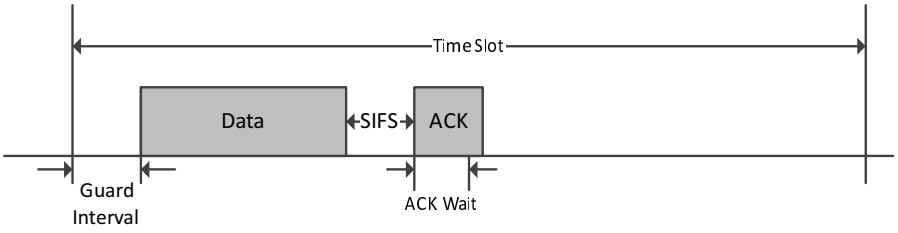


Figure 3.4: RT-WiFi time slot with successful transmission

for the target applications, each RT-WiFi node can only access the channel in its pre-assigned time slots.

Fig. 3.4 illustrates a generic RT-WiFi time slot, and the slot size is determined by Eq. 3.1. The guard interval ($T_{GuardInterval}$) is used to ensure that the consecutive transmissions do not interfere with each other due to synchronization inaccuracy. The guard interval is followed by the data transmission, and the transmission time (T_{Data}) depends on the transmission rate and packet size. When a station successfully receives the data, it will respond with an acknowledgement (ACK) message after a short inter-frame space (SIFS). If sender does not receive ACK after the ACK wait period, the sender assumes that the packet is lost.

$$T_{TimeSlot} \geq T_{GuardInterval} + T_{Data} + T_{SIFS} + T_{ACK} \tag{3.1}$$

Our timer design is based on the Timing Synchronization Function (TSF) in IEEE 802.11 [3]. TSF is achieved by having all nodes keep a local $1MHz$ TSF timer, and all stations synchronize their local timers to the master clock in the RT-WiFi AP through the beacon frames. The beacon frames that contains AP's timing information is distributed by AP periodically. The timer

Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7
AP Broadcast	Shared	STA1 ↓ AP	AP ↓ STA1	STA2 ↓ AP	AP ↓ STA2	STA3 ↓ AP	AP ↓ STA3

Figure 3.5: An example of a superframe with 8 time slots

then triggers accurate timer interrupts to ensure the correct operation of the data link layer. We shall present the implementation details of our timer module on a commonly used Wi-Fi chip in Section 3.3.2.

3.2.2 Link Scheduler

If accurate time synchronization with the master clock in the AP is achieved, every station in the RT-WiFi network achieves consensus on a global time. Based on this global time synchronization, we design a link scheduler to coordinate the channel access among the stations. There are three key data structures maintained in the link scheduler:

Link: A link describes the communication behavior within a time slot. It is specified by its source, destination and the associated link type. We define four link types in RT-WiFi: transmit, receive, broadcast and shared. A broadcast link is used to transmit a data frame to all the neighbor nodes in the network; a transmit link is used for dedicated data frame transmission to the given destination; a receive link is used for receiving a data frame from the given source; and a shared link is for multiple nodes to compete for transmitting data frames.

Superframe: Superframe is a sequence of consecutive time slots. It defines the device's communication pattern with neighbors and it can be repeated infinitely to yield an infinite schedule at run time. Fig. 3.5 gives an example of a superframe configured in an RT-WiFi AP with eight time slots. In this RT-WiFi network three stations are present, and each station is configured with a pair of dedicated transmit and receive link to the AP in the superframe.

Device Profile: Each RT-WiFi nodes keeps the device profile for the centralized network manager to generate superframe and links for each RT-WiFi station. The device profile is gathered when an RT-WiFi station joins the RT-WiFi network, and the device information is forwarded to the network manger via RT-WiFi AP. The network manger installed in the RT-WiFi AP performs access control, construct the link schedule according to the sampling rate requirement of the newly joined device, and then send the superframe and link configuration information back to that station. The network manager design and schedule construction will be discussed in the following sections.

To execute the link schedule specified by the superframe and link configuration, the link scheduler is invoked by every timer interrupt. The link type of the current time slot is determined by looking up the superframe using the timing information. Depending on the link type, the link scheduler either requests a frame to the specified destination from the message handling module, or forwards the received frame to the upper layer. Before passing data frame to the physical layer for transmission, the link scheduler decides the

transmission parameters through the flexible channel access controller which will be described in the following section.

3.2.3 Flexible Data Link Layer Design

To support a wide range of control applications, we need a flexible design in RT-WiFi data link layer. RT-WiFi allows users to choose the following design parameters to achieve the desired application behavior.

3.2.3.1 Sampling rate

Higher sampling rate helps control applications perform more accurate monitoring on the target systems, and thus further improves the quality of control. Increasing the sampling rate, however is challenging because higher sampling rate requires smaller size of the time slot, which is heavily dependent on the following factors.

Packet size and data rate: RT-WiFi cannot successfully transmit a packet if the transmission time of the packet is larger than the time slot size according to Eq. 3.1. IEEE 802.11 specifications allow the use of multiple transmission rates at the physical layer that utilize different modulation and coding schemes. For example, the IEEE 802.11g supports twelve data rates from $1Mbps$ to $54Mbps$. Higher transmission rate provides higher throughput, but it is also less resilient to noise and easy prone to error [42, 52]. Our flexible data link layer design allows the selection of the date rate that best fits the

current channel condition and the desired time slot size.

Computational resource: Increasing the sampling rate of the TDMA data link layer will reduce the time slot size. For executing tasks in a shorter time interval, more computational resource is required. Therefore, the maximum sampling rate supported by an RT-WiFi node is limited by its computation capability.

Synchronization accuracy: The guard interval in Fig. 3.4 is reserved for the drift between two RT-WiFi devices. The minimum slot size is influenced by the synchronization accuracy because the size of time slot has to be larger than the guard interval. We can reduce the guard interval size by utilizing more accurate clock or decreasing the synchronization interval.

3.2.3.2 Reliability

RT-WiFi applies two retransmission mechanisms to improve the reliability of a communication link. These two mechanisms can be used either independently or in combination. Which retransmission mechanism to use depends on the available computation resource and specific control applications.

In-slot retry: The in-slot retransmission mechanism is shown in Fig. 3.6. The retransmission is invoked immediately if the sender does not receive an ACK message from the receiver. The number of retransmission is bounded

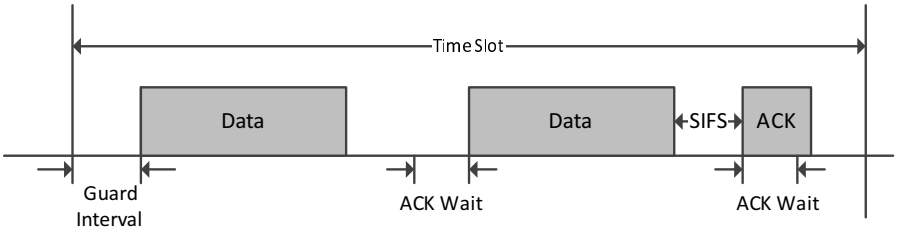


Figure 3.6: RT-WiFi time slot with in-slot retransmission

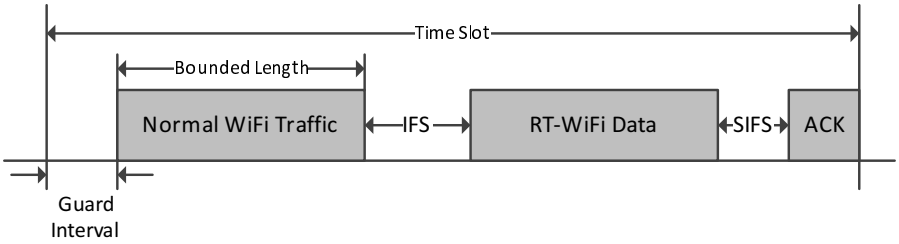


Figure 3.7: RT-WiFi time slot when considering Wi-Fi traffic

by the length of a time slot, since the retransmission time of an in-slot retry should not exceed the length of a time slot.

Out-of-slot retry: If a packet fails to be transmitted in one time slot, RT-WiFi node can be configured to retransmit the lost packet on the next available link. Notice that the retransmission heavily depends on the desired application behavior. For example, on the next available link, if new control/sensor data is available, then it does not make sense to retransmit the old data.

3.2.3.3 Co-existence with regular Wi-Fi network

Due to the pervasive Wi-Fi signal, the RT-WiFi data link layer design needs to consider its co-existence performance with regular Wi-Fi network.

With the presence of Wi-Fi networks in the operation environment, the RT-WiFi data link layer should keep real-time characteristics while minimizing its influence on the surrounding regular Wi-Fi networks as well.

We assume that a regular Wi-Fi node conforms to the standard CSMA/CA channel access mechanism and the maximum consecutive channel access time of a regular Wi-Fi node can be bounded by a constant. This can be achieved by restricting the maximum transmit unit (MTU) size in the MAC layer of Wi-Fi network, and limiting the lowest available transmission speed of that Wi-Fi network. This assumption must be enforced in our scheme because if any Wi-Fi node (e.g., a jammer) can occupy a channel and continuously transmit data for an arbitrarily long time then there is no way to achieve real-time data delivery. If this assumption is valid, then RT-WiFi can use the following two mechanisms to improve its co-existence performance with regular Wi-Fi networks.

Performing Carrier Sense: Without the presence of Wi-Fi traffic in the operation environment, an RT-WiFi node does not perform carrier sense on its assigned links before the transmission. This is because the network manager will make sure that there is no temporal or spatial reuse in the operation environment on the channel specified in that time slot. However, if there are regular Wi-Fi networks operating in the surrounding environment, the co-existence performance between RT-WiFi and regular Wi-Fi could be poor because of the high possibility that the two types of traffic could collide with each other. To

avoid this situation, RT-WiFi nodes should perform clear channel assessment (CCA) and only start the transmission when the channel is clear.

Shortening Inter-frame Spacing (IFS): IFS is an interval defined between the transmission of two consecutive Wi-Fi packets. Wi-Fi nodes are required to wait for a pre-defined IFS before they can start to transmit the next frame. To grant a higher priority for the RT-WiFi node to access the channel, a shorter IFS can be assigned for the RT-WiFi node, so that they can start transmission earlier than other regular Wi-Fi nodes.

Fig. 3.7 illustrates the time slot when we consider the co-existence of both Wi-Fi and RT-WiFi traffic. The real-time data delivery still can be achieved because RT-WiFi only waits for a bounded time, and RT-WiFi node has a higher priority to access a channel. The cost for improving the co-existence performance is to delay the data transmission of RT-WiFi traffic. However, we expect larger jitter in the sensor and control data as the price to pay for better co-existence performance, as it also limits the highest supported sampling rate because we need to reserve blocking time from Wi-Fi traffic.

In this subsection, we only discuss the available co-existence mechanisms in the RT-WiFi MAC layer. While the proposed mechanisms can achieve predictable data delivery, they lead to inefficient channel usage due to pessimistic estimation of the worst case transmission time of regular Wi-Fi traffic. To alleviate the inefficient channel usage, we will propose the design of an efficient co-existence scheme in the network manager in Section 5.4.

3.3 RT-WiFi MAC Layer System Implementation

In this section, we present the implementation details of the RT-WiFi MAC layer protocol on commercial-off-the-shelf Wi-Fi chips. We first summarize our hardware and software platform, and then describe the challenges and our solutions when we implement the key components of the RT-WiFi protocol.

3.3.1 Hardware and Software Platform

RT-WiFi does not rely on specific IEEE 802.11 hardware implementation, and our design should be able to port to any IEEE 802.11 compatible hardware. We have developed RT-WiFi MAC layer protocol on top of Qualcomm Atheros AR9200 family Wi-Fi chips. We choose Qualcomm Atheros AR9200 Wi-Fi chips in our implementation because they are one of the most commonly used Wi-Fi chips and they have open source driver module in Linux operating system. We have successfully tested our implementation on Qualcomm Atheros AR9280 and AR9285 Wi-Fi chips. In the following, we use AR9285 as an example to present our MAC layer implementation. Operating in the $2.4GHz$ frequency band, AR9285 Wi-Fi chip supports IEEE 80211 b/g/n specifications and its maximum transmission rate is $150Mbps$.

We use Ubuntu 14.04 as the operating system, which runs Linux kernel 3.13.0. On top of that, we adopt open source Linux compat-wireless driver [8] as the foundation of RT-WiFi. Two software modules that originate from compat-wireless driver, mac80211 and hardware-dependent driver module, are

incorporated with the TDMA design to build the MAC layer of RT-WiFi. The mac80211 is a framework for developing software-based MAC driver; it is used for supporting the IEEE 802.11 frame management and serves as the interface between upper network layer and the message handling module of RT-WiFi MAC layer. We use hardware driver module ath9k for supporting AR9285, and a software interface is implemented for message handling and flexible channel access configuration.

3.3.2 Timer Module

The basic time unit in a RT-WiFi network is a time slot. Depending on the adopted IEEE 802.11 physical layer, the size of the time slot can be estimated by Eq. 3.1. In the following, we give a concrete example by using IEEE 802.11g with ERP-OFDM modulation scheme. $T_{GuardInterval}$ depends on the accuracy of timing synchronization, which relies on the Timing Synchronization Function (TSF) in IEEE 802.11. According to the IEEE 802.11 standard [3], the drift rate of TSF should be within $\pm 0.01\%$. Users can control the maximum amount of drift between AP and stations by controlling the broadcasting interval of TSF. For example if the beacon is broadcast every $100ms$, then the maximum drift is within $20\mu s$. To determine the value of T_{Data}, we assume that the application payload is 200 bytes, UDP header is 8 bytes, and IPv4 header is 20 bytes. The MAC header is 32 bytes with 4 bytes frame check sequence. According to [76], the physical layer convergence procedure (PLCP) preamble and header delay are $20\mu s$. If the data is transmitted

with the maximum data rate (54Mbps), then

$$T_{Data} = 20 + \frac{(200 + 8 + 20 + 32 + 4) \cdot 8}{54 \cdot 10^6} = 59.11\mu s. \tag{3.2}$$

T_{SIFS} is $10\mu s$ for IEEE 802.11g. The total size of ACK frame is 14 bytes. Similarly, we can derive T_{ACK} as following.

$$T_{ACK} = 20 + \frac{14 \cdot 8}{54 \cdot 10^6} = 22.07\mu s. \tag{3.3}$$

Theoretically the minimum slot size based on aforementioned settings is

$$T_{TimeSlot} \geq 20 + 59.11 + 10 + 22.07 = 111.18\mu s. \tag{3.4}$$

The size of the time slot supporting in-slot retry and co-existence with regular Wi-Fi can be calculated in a similar manner.

Based on TSF, the timer module in the data link layer sets its local hardware timer in AR9285 and triggers the timer interrupt at the beginning of a time slot. Originally, the handler of this timer interrupt is deferred and handled by a kernel tasklet later. In our implementation, however we observe that this tasklet could be blocked by other tasklet that has the same priority, and this delays the execution of time sensitive RT-WiFi task. To avoid this delay and meet the stringent timing requirement of RT-WiFi, instead of deferring the execution of RT-WiFi task to tasklet we directly execute it in the interrupt service routine. Notice that this design reduces the delay for executing RT-WiFi task at the cost of increasing kernel response time. We note that the average execution time of RT-WiFi task for Intel Atom N450 CPU is less than $20\mu s$ which has small impact on the kernel response time.

3.3.3 Link Scheduler and Message Handling Module

To implement TDMA-based channel access, we build two message handling modules in RT-WiFi data link layer for sending and receiving packets. To ensure efficient message handling, the message transmission module consists of multiple queues, one for each dedicated real-time communication link and one for non-real-time communication link. The message transmission module receives packets from mac80211 module, and is controlled by the link scheduler to transmit a packet to the lower layer at its assigned time slot. The message reception module enqueues messages and forwards them to upper layer whenever it gets a message.

Both real-time and non-real-time traffic can be present in an RT-WiFi node. In order to identify the traffic type in the data link layer, we utilize the type of service (TOS) field in the IP header. Application developer can create a socket with specified TOS, by using user level API, *i.e.,* setsockopt(). The mac80211 module will map TOS to an access class. In our current implementation, RT-WiFi maps the highest priority class to RT-WiFi queues. Messages in that class will be sent on dedicated transmit links. For non-real-time packets, the link scheduler will transmit them on shared links.

After the RT-WiFi node associates with the AP, its link scheduler will configure the timer interrupt, and start to transmit and receive messages according to the configured link schedule.

3.3.4 Flexible Channel Access Controller

To achieve adaptable channel access, we implement a flexible channel access controller in RT-WiFi MAC layer. The controller configures the hardware setting according to the user specified parameters. Changing the sampling rate is achieved by modifying the hardware timer. To control the reliability level, we manipulate the retry chains and communication schedules as follows. AR9285 has a feature called retry chains (*rates* in *struct ath_tx_info*), which can be used to set a series of in-slot retries with corresponding transmission rates. If an out-of-slot retry is enabled, RT-WiFi will confirm the delivery of a frame by checking the reception of the corresponding ACK message. If the ACK message is missing, RT-WiFi will re-queue that frame into the transmission queue.

We also utilize several hardware features to control channel access and co-existence mode. To control carrier sensing in a channel, we configure the hardware register of AR9285 (AR_DIAG_SW) to enable or disable the physical and virtual carrier sensing. To disable random back-off mechanism of a hardware queue, we set the contention windows size to zero for the transmission queue, and disable the post frame back-off by configuring hardware register *AR_DMISC*. AR9285 has eight hardware queues, and we map the RT-WiFi traffic to one of the hardware queues. To enable the co-existence mode with regular Wi-Fi network, we manipulate the hardware register (AR_DIAG_SW) to enable carrier sensing. To configure the size of IFS, we can set the IFS of the hardware queue by configuring register (AR_DLCL_IFS), which lets RT-WiFi

traffic have a higher priority to access the channel.

3.4 Performance Evaluation

To validate our design of the RT-WiFi MAC layer protocol and evaluate its performance in providing high-sampling-rate real-time wireless communication, we set up a testbed for the performance comparison between RT-WiFi and regular Wi-Fi networks. This testbed consists of one AP and three stations (STA1, STA2 and STA3) which form a star network topology. All the four devices are equipped with the same Atheros AR9285 IEEE 802.11 compatible wireless interface but different CPUs with varied computation power. This setting demonstrates that the implementation of RT-WiFi only introduces limited computation overhead, and our implementation runs smoothly on resource-constrained embedded devices.

To emulate the sensing and control flows in a wireless control system, we install a UDP socket program on each device. Sensor data with a fixed size payload are transmitted from each station to the AP according to the configured sampling rate. On the other direction, the AP periodically transmits control data with the same packet size back to each station. Notice that sensors, actuators, and controllers are all running in the application layer. The overall delay from a sensor to a controller or from a controller to an actuator includes the application layer to MAC layer delay in each device and the MAC layer to MAC layer delay between the devices. In this section we focus on evaluating the MAC layer performance. We will evaluate the application

layer performance in Section 6.3.

We compare the MAC layer to MAC layer performance between RT-WiFi and regular Wi-Fi in two test scenarios, an interference-free environment and an office environment with Wi-Fi traffic. The main performance metrics we used in the experiments include the data link layer transmission latency and the packet loss ratio. The data link layer transmission latency is calculated as the difference of a frame's TSF timestamps between the receiver side and the sender side; The packet loss ratio measures the percentage of packets lost by tracking the sequence number of each packet.

The following configuration is used in our experiments. We run each experiment for 600 seconds. Both the RT-WiFi and regular Wi-Fi are configured to use channel 6 of IEEE 802.11 b/g/n. There are in total six pairs of UDP sockets in the experiment setting between the stations and the AP (STA1→AP, STA2→AP, STA3→AP, AP→STA1, AP→STA2 and AP→STA3). For each UDP socket, we publish the data every $4ms$. The application layer payload is fixed at 460 bytes. For the RT-WiFi network, we set the time slot size as $500\mu s$, and we use the same superframe and link schedule as shown in Fig. 3.5. The data rate is set as $54Mbps$.

3.4.1 Performance Evaluation in Interference-free Environment

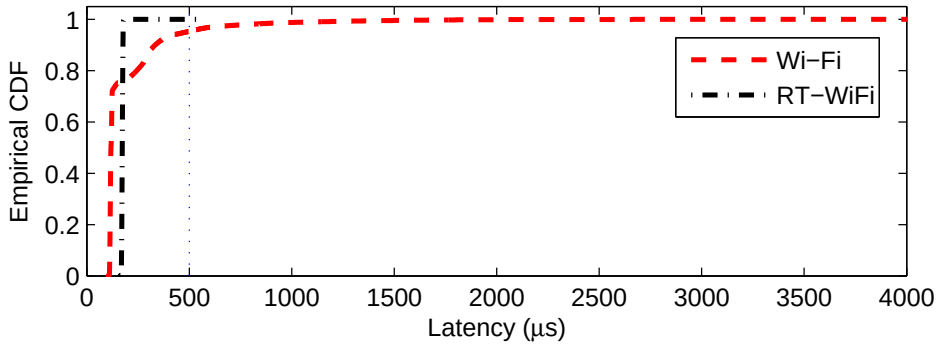
To create an interference-free environment for performance comparison, we deploy the testbed in a parking lot of a hiking trail in Austin city suburban area around Texas State Highway Loop 360 and Loop 1. Table 3.1 summa-

Link	Max Delay (μs)		Mean Delay (μs)		Delay Standard Deviation (μs)		Loss Ratio	
	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi
STA1→AP	535	16448	173	191	4.88	231.60	0.19%	0%
STA2→AP	529	13387	172	181	2.52	214.15	0.16%	0%
STA3→AP	525	13589	174	202	3.01	236.75	0.21%	0%
AP→STA1	827	16472	184	250	5.29	342.24	0.06%	0%
AP→STA2	544	17465	187	298	3.98	360.66	0.04%	0%
AP→STA3	1055	17049	188	248	4.87	325.50	0.01%	0%

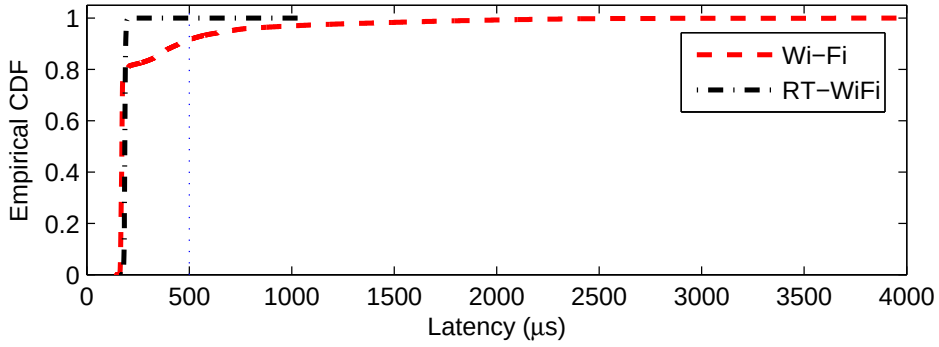
Table 3.1: Latency and packet loss ratio comparison in an interference-free environment

rizes the comparison results between RT-WiFi and regular Wi-Fi in this test scenario. The results include the max, mean and standard deviation of the MAC layer to MAC layer transmission delay, and the packet loss ratio for each of the six links. The empirical cumulative density functions (ECDFs) of the transmission latency for uplinks (STAs to AP) and downlinks (AP to STAs) are presented in Figure 3.8. We observed from the experimental results that RT-WiFi protocol can provide accurate real-time packet delivery in that more than 99.98% of packets are delivered within the assigned $500\mu s$ time slot and the standard deviation of the latency in RT-WiFi network is less than $5.3\mu s$. The occasional packet loss is caused by the interference from other visitors' mobile devices. This is verified by observing uncontrolled probe request frames from unknown devices to RT-WiFi AP when visitors are nearby.

As can be observed in Table 3.1, regular Wi-Fi network has a higher average delay and a larger transmission variation. This is because four nodes in the Wi-Fi network compete for media access by utilizing CSMA/CA and



(a) Latency ECDF for uplinks (STAs→AP)



(b) Latency ECDF for downlinks (AP→STAs)

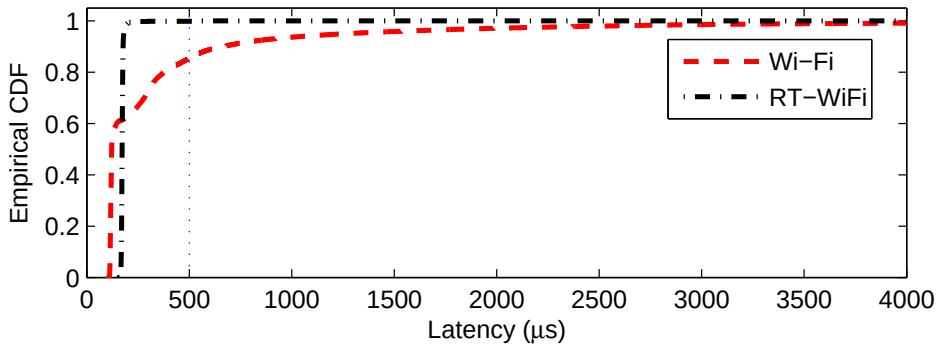
Figure 3.8: MAC layer transmission latency comparison between Wi-Fi and RT-WiFi in an interference-free environment

random back-off mechanisms. The mean delay of downlinks are higher than that of uplinks is because the packets sent from AP not only have to wait for channel access but also are delayed due to the queuing effect. RT-WiFi network is seldom affected by the queuing effect. This is because to delay a packet transmission in the AP, one packet needs to be deferred for more than two time slots ($1000\mu s$) according to our configured link schedule. As shown in Fig. 3.8, only less than 0.01% packets in the RT-WiFi network has latency larger than $1000\mu s$. The latency variation in Wi-Fi network is up to 90 times to that of RT-WiFi network, and the maximum delay is more than 30 times to RT-WiFi, which poses great challenges for the controller design.

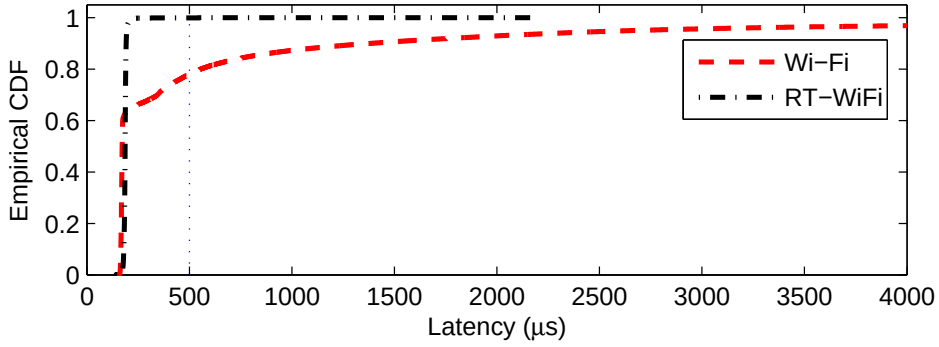
Another observation from Fig. 3.8a is that the minimum latency of RT-WiFi on the uplinks is higher than that of regular Wi-Fi. This is due to the delay of the TDMA guard interval and the implementation overhead in the MAC layer software module. However, as Fig. 3.8b shows, the difference of the minimum latency between regular Wi-Fi and RT-WiFi is smaller for the downlinks. This is because the minimum latency of regular Wi-Fi is increased by queuing delay.

3.4.2 Performance Evaluation in Office Environment

To evaluate the performance of the RT-WiFi network in the presence of interference, we deploy the testbed on the 5th floor of the GDC building in the University of Texas at Austin. This floor is covered with more than 10 Wi-Fi APs, which spread to all the orthogonal Wi-Fi channels in ISM 2.4GHz band.



(a) Latency ECDF for uplinks (STAs→AP)



(b) Latency ECDF for downlinks (AP→STAs)

Figure 3.9: MAC layer transmission latency comparison between Wi-Fi and RT-WiFi in an office environment

Under this setting, we repeat the same experiments as in the interference-free environment.

Fig. 3.9 compares the MAC layer latency between RT-WiFi and Wi-Fi network in the office environment. Compared with the interference-free environment, the latency of regular Wi-Fi network is increased because the office environment has more inference from existing Wi-Fi networks. Thus, the CSMA/CA and random backoff mechanisms have a higher chance to defer

Link	Max Delay (μs)		Mean Delay (μs)		Delay Standard Deviation (μs)		Loss Ratio	
	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi	RT-WiFi	Wi-Fi
STA1→AP	3865	100078	176	401	25.86	1491.69	8.64%	0%
STA2→AP	4193	81499	171	348	27.62	1000.60	9.97%	0%
STA3→AP	3861	75298	174	429	25.16	1221.72	7.55%	0%
AP→STA1	1197	78089	184	788	16.86	2861.42	9.52%	0%
AP→STA2	1342	78923	189	790	15.19	2806.56	8.09%	0%
AP→STA3	2186	77860	189	799	19.03	2855.89	9.31%	0%

Table 3.2: Latency and packet loss ratio comparison in an office environment

the transmission of regular Wi-Fi packets. The latency distribution of RT-WiFi network in Fig. 3.9 is similar to that in Fig. 3.8, because we adopt the same TDMA design, which targets to provide small latency variation.

Table 3.2 shows the latency and packet loss ratio of RT-WiFi network and Wi-Fi network. The maximum latency of RT-WiFi network is increased up to $4.2ms$ because there are more uncontrolled mobile devices around in the office environment. The packet loss ratio of RT-WiFi network increases to 10% due to the collision with background traffic. However, because of the TDMA design of RT-WiFi network, the mean delay in the office environment remains similar to the interference-free environment, which makes it convenient for the control designer to reuse their controller design. Moreover, the delay variation of RT-WiFi is still small (less than $30\mu s$), which makes it easy for control applications to compensate the packet loss.

By design, Wi-Fi network can achieve zero packet loss because it keeps retransmitting until the data is delivered. However, retransmitting out-of-date

	Max Delay (μs)	Mean Delay (μs)	Delay Standard Deviation (μs)	Loss Ratio
Regular Wi-Fi	62629	580	1679.04	0%
RT-WiFi baseline	953	183	27.75	50.21%
RT-WiFi co-ex	2507	220	149.27	10.92%
RT-WiFi co-ex + retry	2831	254	166.37	4.96%

Table 3.3: Latency and packet loss ratio comparison with present of large network traffic

sensor data does not provide useful state information if new sensor data is already available, and the retransmission may further delay the transmission of new data. According to Table 3.2 the maximum delay of regular Wi-Fi network is up to $100ms$, and the standard deviation is increased to $2800\mu s$. The large delay of sensor data will make the controller lose track of the system state, and large latency for delivering control data could also cause the target system to be out of control. More importantly, the high latency variation in Wi-Fi networks poses serious challenges to controller design.

3.4.3 Flexible Channel Access Controller

In order to provide a guideline for users to adjust design parameters in the flexible channel access controller, we evaluate the performance of the flexible channel access controller under various settings.

Reliability and Co-existence with regular Wi-Fi Network To demonstrate the capability of adjusting the reliability and co-existence with regular Wi-Fi network settings in the flexible channel access controller, we set up a testbed in the same office environment with two networks as shown in Fig. 3.10.

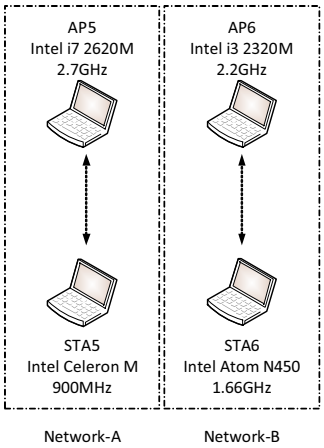


Figure 3.10: Flexible channel access control testbed setting

Network-A is a regular Wi-Fi network, which utilizes network traffic generator, iperf [6], to generate a $10Mbps$ UDP traffic from STA5 to AP5. The MAC layer maximum transmission unit of Wi-Fi network is configured as 300 bytes, for bounding the packet transmission time. For Network-B, we run the same UDP socket program, which publishes data every $4ms$ from AP6 to STA6. We configured Network-B by using four settings: regular Wi-Fi, RT-WiFi baseline, RT-WiFi with co-existence, RT-WiFi with co-existence and one in-slot-retry; we record the latency and packet loss ratio for link (STA6→AP6).

We present the experimental results in Table 3.3. Because of the interference from Network-A, the mean delay of regular Wi-Fi is increased to $580\mu s$, which is 47% higher than the average mean delay of the upload links in regular Wi-Fi network as shown in Table 3.2. The packet loss ratio of baseline RT-WiFi network is also increased dramatically to 50.21% due to the large

Sampling Rate (Hz)	Max Delay (μs)	Mean Delay (μs)	Delay Standard Deviation (μs)	Loss Ratio
1000	1792	125	25.78	3.71%
2000	1639	125	22.47	7.29%
4000	1850	127	24.69	8.75%
6000	1460	122	21.56	13.62%

Table 3.4: Latency and packet loss ratio of RT-WiFi with different sampling rates in an office environment

interference from Network-A. For RT-WiFi network in the co-existence mode, the packet loss ratio is decreased to 10.92% because we enable the carrier sense mechanism. Carrier sense cannot eliminate all packet loss because of the hidden terminal problem. The mean delay of Wi-Fi network with co-existence mode is increased to $230\mu s$, and the standard deviation of its delay is increased to $149.27\mu s$. This is due to the transmission of RT-WiFi packets which is delayed by the interference from Network-A while the carrier sense mechanism is enabled. The packet loss ratio is further decreased to 4.96% when we enable the in-slot-retry, which also increases the mean delay and delay standard deviation.

Sampling Rates For demonstrating the capability of adjusting the sampling rate in the flexible channel access controller, we adjust the sampling rate of a RT-WiFi network from $1kHz$ to $6kHz$. We set up a testbed with one AP and one station in the office environment, and we generate one way UDP traffic from the station to the AP by iperf. Since the size of a time slot in $6kHz$ sampling rate is only $166\mu s$, the application layer payload of the UDP traffic is fixed to 100 bytes. In practice, we use a 8-bytes floating point

number to represent one degree of freedom in a control system. One 100 bytes of payload is enough to control a dexterous robot hand with 12 degrees of freedom. Moreover, as physical layer technology advances, the RT-WiFi system will be able to transmit more data per time slot.

We measure the latency and packet loss ratio from the station to the AP. Table 3.4 summaries our experimental result. As shown in Table 3.4, the measurement of delay in different sampling rates are similar because the packet size and data rate for this experiment is fixed. We demonstrate that our implementation is capable to operate in $6kHz$, since the latency measurement in $6kHz$ remains similar to other lower sampling rates. The mean latency in Table 3.4 is around 30% less than the mean latency in Table 3.2. It is due to smaller application payload in this experiment. The loss ratio in Table 3.4 is increased as we raise the sampling rate. It is because in this experiment we do not enable the co-existence mode, and the generated UDP traffic occupies channel longer when we increase the sampling rate. Therefore, traffic with higher sampling rate is more likely to collide with surrounding transmissions, which results in higher loss ratio.

Chapter 4

Jitter-Free Scheduling in RT-WiFi Networks

Although RT-WiFi data link layer can provide predictable packet delivery, how to construct the data link layer communication schedule is not specified and left to the system designer to exploit for satisfying other requirements. In this chapter, we consider the network resource scheduling problem, and discuss how to construct a communication schedule that is not only feasible but also has the transmission jitter minimized for every RT-WiFi device. We firstly present the background information in Section 4.1, and describe the communication model in Section 4.2. Section 4.3 formalizes our target schedule assignment problem, and presents the algorithms under static configurations. Section 4.4 further extends the algorithm to address schedule assignment with network dynamics. We summarize the experimental results in Section 4.5.

This chapter is previously published in [54]. I contributed algorithm design and implementation to this published work.

4.1 Introduction

In the previous chapter, we presented the design and implementation of RT-WiFi MAC layer that provides flexible real-time high-speed wireless communication, and supports predictable timing guarantee on packet delivery. While guaranteeing packet delivery latency is an essential step in achieving networked control of wireless cyber-physical applications, how to construct the data link layer communication schedule that meet the required communication requirements is not specified. While the constructed communication schedule shall fulfill the sampling rate requirements for the sensing and control devices, the approach we build the schedule also determines the communication jitter [30, 59] that can significantly impact system performance.

One of a worth mentioning feature of networked control systems is that the device sampling rate is usually not a fixed value, but within a given range. This feature allows the system designer to choose the most appropriate sampling rate to achieve the best performance in specific applications. The lowest allowed sampling rate generally reflects the required performance while the highest supported rate is constrained by the hardware limitation. Thus, we consider the network resource scheduling problem that the design goal of our scheduling algorithm is to choose an appropriate sampling rate for each sensing or control device, and to construct a communication schedule that is not only feasible but also has the transmission jitter minimized for every device.

Besides the communication jitter, a complicating factor to this sched-

ule construction task is that the network devices may join/leave the system either actively or passively, and may update their requests for communication resource during system operation. How to adapt to these system dynamics while still maintaining the communication schedule with minimized jitter is a challenging issue. Most existing works in the literature, unfortunately, are based on the common assumption that the network topology of the wireless control system under study is static and the devices' network resource requirements remain unchanged. These algorithms thus cannot be directly applied to our problem.

As a solution, we shall present an effective data link layer schedule assignment algorithm for RT-WiFi networks to minimize the jitter for every device in the network. We start with considering a static network topology, and formalize it as a real-time scheduling problem under the periodic task model to find an appropriate sampling period and transmission phasing for each device. To deal with network dynamics, we further introduce a novel data structure called S-tree to represent the communication schedule, and propose an efficient algorithm for dynamic network resource allocation while minimizing the overhead associated with the schedule adjustment. Furthermore, we incorporate the proposed data link layer scheduling algorithms in RT-WiFi network manager. We will present the design and implementation of RT-WiFi network manager in Chapter 6.

4.2 RT-WiFi Data Link Layer Communication Model

The communication links in a wireless networked control system can be categorized into two types: sensing links to deliver sensor measurements, and control links to send control messages to the actuators. These communication links usually deliver data in a periodic manner. The period of a communication link is usually not a fixed value but within a range. It is decided by its communication resource requirement, hardware and software limitations, and the current network capacity. For this reason, in this chapter, we model the communication links in an RT-WiFi network as a set of periodic communication links $L = \{L_i\}_{i=1}^{n}$. Each link L_i is defined as a triple $L_i = (P_i^{min}, P_i^{max}, c_i)$, and consists of an infinite sequence $\mathcal{L}_{i,j}(j = 0, 1, \dots)$. In this model, P_i^{min}, P_i^{max} and c_i are all positive integers, representing the number of time slots. In terms of scheduling problems, the scheduler allows preemption only at integral time slot boundaries, absent other application constraints.

In this link definition, P_i^{min} and P_i^{max} represent the minimum and maximum supported sampling period of a communication link respectively. The control system designer specifies the maximum allowed sampling period (*i.e.*, the minimum required sampling rate) for satisfying the control performance requirements of the system. On the other hand, the minimum sampling period a device can achieve is limited by its hardware and software capability. To fulfill the communication requirements of a networked control system, *i.e.*, all the periodic communication links are schedulable, each link L_i should be assigned an integer period P_i properly, with $P_i^{min} \leq P_i \leq P_i^{max}$.

The communication time c_i is defined as the number of time slots needed for transmitting the sensor or control data. Depending on the maximum payload size of a packet in the RT-WiFi network, the sensor or control data may need to be divided into multiple fragments for transmission. A link with multiple fragments for transmission is called a multi-fragment link. We define $F_{i,j}$, the completion time of $L_{i,j}$ as the finish time of its last fragment. Thus the inter-completion time between $L_{i,j}$ and $L_{i,j-1}$, $I_{i,j}$, is $F_{i,j} - F_{i,j-1}$. We further define network utilization and communication jitter as following.

Definition 1. *Given a set of communication links $L = \{L_i\}_{i=1}^n$, the utilization of the network is $U = \sum_{i=1}^n \frac{c_i}{P_i}$.*

Definition 2. *Given an inter-completion time series $\{I_{i,1}, \cdots, I_{i,n}\}$ of link L_i, the jitter of L_i is defined as the average variation of packets inter-completion times, which is $\frac{\sum_{k=2}^n (I_{i,k} - I_{i,k-1})^2}{n-1}$.*

To achieve good control performance, the constructed schedule should eliminate or minimize the jitters for the communication links. In the following of this chapter, we will present efficient algorithms for constructing feasible communication schedule in RT-WiFi networks with minimum jitters, in both static and dynamic network setup.

4.3 Static Link Schedule Assignment in RT-WiFi Networks

In this section, we study the data link layer schedule assignment in RT-WiFi networks with static network setup. We assume that the network topology and link characteristics are obtained from a topology learning phase and do not change afterwards. We focus on constructing a jitter-free communication schedule, in which the periods selected for the communication links form a harmonic chain and the network utilization is minimized. Static schedule assignment algorithm is especially useful for offline scheduling analysis in the controller design phase. We will study the dynamic schedule assignment and adjustment in RT-WiFi networks in Section 4.4.

4.3.1 Jitter-Free Scheduling Problem

In static RT-WiFi networks, we consider to construct a jitter-free schedule for a set of communication links. We first describe the formal definition of the jitter-free scheduling problem, and then discuss the complexity of this problem.

Definition 3. *Jitter-Free Scheduling (JFS) Problem: Given a communication link set* $L = \{L_i\}_{i=1}^{n}$*, with* $L_i = (P_i^{min}, P_i^{max}, c_i)$*, the JFS problem is to determine the period* P_i *and a series of phasing* $\{\phi_{i,1}, \cdots, \phi_{i,c_i}\}$ *(*$\phi_{i,j}$ *is an integer, in number of time slots) for link* L_i*, so that: (1)* $P_i \in [P_i^{min}, P_i^{max}]$*; (2) each fragment of* L_i *is scheduled consecutively at time slot* $(\phi_{i,j} + P_i \cdot k)$*, where* $1 \leq j \leq c_i$ *and* $k = 0, 1, 2, 3, \ldots$*; (3) only one fragment is scheduled at a time*

slot; and (4) the network utilization $U = \sum \frac{c_i}{P_i}$ *is minimized.*

We prove that the jitter-free scheduling (JFS) problem is NP-hard by showing that the decision version of the JFS problem is NP-hard. The decision version of JFS only considers whether a given instance of the JFS problem is schedulable (without considering constraint (4) in Definition 3).

Theorem 1. *To decide if an instance of the JFS problem is schedulable or not is NP-Hard.*

Proof. To show that the JFS problem is NP-Hard, we reduce the one server periodic maintenance problem (PMP_1) in [60] into the JFS decision problem. As shown in [60], the k-server periodic maintenance problem is NP-Complete in the strong sense if $k \geq 1$. The input of PMP_1 problem is as following. Given a multiset of positive integers $A = \{m_1, m_2, \cdots, m_n\}$, where m_i represents the maintenance period of machine i. An one server schedule S for A is an infinite sequence of one multiset over $\{1, 2, \cdots, n, blank\}$, such that successive occurrences of i are exactly m_i slots apart.

We reduce an instance of PMP_1 problem into the JFS decision problem as follows. For each element m_i in A, we construct a communication link L_i where $c_i = 1$ and $P_i^{max} = P_i^{min} = m_i$. This reduction can be done in polynomial time, since this reduction takes $O(1)$ time for each element in A.

If there exists a solution in PMP_1 problem, then we can set $\phi_{i,1}$ of link L_i to position of the first occurrence of i in the PMP_1 schedule. Because the

successive occurrences of i are exactly m_i slots apart and $P_i^{max} = P_i^{min}$, the period and phasing assignment is a valid solution for the JFS problem. On the other hand, if there is a solution for the JFS problem, we can construct a schedule for PMP_1 such that the occurrences of i is at $(\phi_{i,1} + P_i \cdot k)$, where $k = 1, 2, 3, \cdots$. We fill *blank* to the rest of schedule if no link is scheduled at that time. This is a valid schedule for PMP_1 because JFS makes sure that only one fragment is scheduled at a particular time slot, and the occurrence of the successive fragments are exactly P_i time slot apart. Therefore, there also exists a solution for the PMP_1 problem. This completes the proof. □

4.3.2 Harmonic Chain Based Jitter-Free (HCJF) Scheduler

Since the JFS problem is NP-Hard, we are more interested in finding a sufficient condition for the JFS problem such that the proposed solution is in polynomial time. In the following, we propose a two-stage solution to solve the JFS problem. In the first stage called *period selection*, we present an algorithm to decide a proper period for each communication link, and their periods can form a harmonic chain with minimum network utilization. Based on the selected periods, in the second stage called *phasing assignment*, we further present an algorithm to assign the phasing for each link and construct a communication schedule where no jitter will present. We shall first give the definition of a harmonic chain and then elaborate the two stages in the following subsections.

Definition 4. *A set S of positive integers is a harmonic chain if and only if*

$\forall x, y \in S, (x \mid y) \vee (y \mid x)$. *$x \mid y$ denotes x divides y, or y is a multiple of x.*

To show the benefit of using harmonic chain based period selection, we first give a motivating example by comparing harmonic chain base period selection with the state-of-the-art solution for the jitter-free scheduling problem in wireless network environment. [20] proposes a contention-free (CF) periodic message scheduler, which minimizes the jitter by selecting periods for links so that their periods are all in power of two. Specifically, given a link with its maximum period P^{max}, CF selects link period P to be $2^{\lfloor log_2 P^{max} \rfloor}$. The drawback of CF is that it does not explore the whole period range $[P^{min}, P^{max}]$, which may choose unnecessarily small periods for the links, thus increase the network utilization dramatically. For example, given three communication links $L_1 = (2, 15, 1)$, $L_2 = (10, 30, 1)$ and $L_3 = (10, 60, 1)$, CF assigns their periods P_1, P_2 and P_3 to be 8, 16 and 32 respectively, and the network utilization for the three links is 0.219. On the other hand, our harmonic chain based jitter-free (HCJF) scheduler can choose the periods to be 15, 30 and 60 respectively. This leads to a network utilization of 0.117, which is 46.6% lower than the utilization from CF scheduler. Our proposed HCJF can always outperform CF because CF only explores periods in power of two, which is a special case of harmonic chain.

4.3.2.1 Period Selection

The period selection stage decides how to select the period P_i for each link L_i within its period range $[P_i^{min}, P_i^{max}]$, so that all selected periods form

Algorithm 1: Period Selection Algorithm

Input: A link set $L = \{L_i\}_{i=1}^{n}$ with $L_i = (P_i^{min}, P_i^{max}, c_i)$.
Output: an array $s[]$ of the selected period for each link.

```
// Initialization
```
1 **for** $i \leftarrow 1$ ***to*** n **do**
2 **for** $j \leftarrow 0$ ***to*** $P_i^{max} - P_i^{min}$ **do**
3 $p[i][j] = P_i^{min} + j$
4 **for** $i \leftarrow 2$ ***to*** n **do**
5 **for** $j \leftarrow 1$ ***to*** $p[i].length$ **do**
6 $prev[i][j] \leftarrow null$
7 $u[i][j] \leftarrow \infty$
8 **for** $j \leftarrow 1$ ***to*** $p[1].length$ **do**
9 $prev[1][j] \leftarrow 0$
10 $u[1][j] \leftarrow \frac{c_1}{p[1][j]}$

```
// Compute and memorize subproblem solutions
```
11 **for** $i \leftarrow 2$ ***to*** n **do**
12 **for** $j \leftarrow 1$ ***to*** $p[i].length$ **do**
13 **for** $k \leftarrow 1$ ***to*** $p[i-1].length$ **do**
14 **if** $prev[i-1][k] \neq null$ ***and*** $p[i-1][k] \mid p[i][j]$ ***and*** $\frac{c_i}{p[i][j]} + u[i-1][k] < u[i][j]$ **then**
15 $u[i][j] \leftarrow \frac{c_i}{p[i][j]} + u[i-1][k]$
16 $prev[i][j] \leftarrow k$

```
// Retrieve result
```
17 $k = \arg\min_k u[n][k]$
18 **if** $prev[n][k] == null$ **then**
19 **return** *null*
20 **for** $i \leftarrow n$ ***to*** 1 **do**
21 $s[i] \leftarrow p[i][k]$
22 $k \leftarrow prev[i][k]$
23 **return** s

a harmonic chain and the network utilization is minimized. By controlling link periods to form a harmonic chain, we can eliminate communication jitters with proper phasing assignments. We formalize the period selection problem as follows:

Definition 5. *Period Selection Problem: Given a communication link set* $L = \{L_i\}_{i=1}^{n}$*, with* $L_i = (P_i^{min}, P_i^{max}, c_i)$*. Without loss of generality, we assume that the links are sorted in the non-decreasing order of their maximum allowed period,* i.e.*,* $P_i^{max} \geq P_j^{max}$ *for* $i > j$*. Ties are resolved in favor of the link with a larger minimum allowed period. The period selection problem is to find* P_i *for every link* L_i*, so that 1)* $P_i \in [P_i^{min}, P_i^{max}]$*; 2)* $\{P_i\}_{i=1}^{n}$ *form a harmonic chain, and 3) the network utilization* $U = \sum_{i=1}^{n} \frac{c_i}{P_i}$ *is minimized.*

We define a *configuration* Θ as an assignment of period P_i for every link L_i in L. A *valid configuration* is a configuration whose periods form a harmonic chain. Suppose $P_{i,j}$ is the j^{th} admissible period of link L_i, $P_i^{min} \leq P_{i,j} \leq P_i^{max}$, a *partial configuration* $\Theta(P_{i,j})$ is an assignment of period $\{P_1 \cdots P_i\}$, where $P_i = P_{i,j}$. We use the network utilization to measure the quality of a valid configuration. The goal of the period selection algorithm is to find the optimal valid configuration, that is the valid configuration with the minimum network utilization.

Lemma 1. *The optimal valid partial configuration* $\Theta(P_{i,j})$ *has a non-decreasing property,* i.e.*,* $\forall m < k, P_m \mid P_k$*.*

Proof. We prove this lemma by contradiction. Suppose that there exists an optimal valid partial configuration $\Theta_{opt}(P_{i,j})$ which does not hold the non-decreasing property. Assuming that in $\Theta_{opt}(P_{i,j})$, u is the first place where the configuration decreases, *i.e.*, $u < v$ and $P_u > P_v$. In this case, we can increase P_v to P_u to get a better configuration, because 1) the new period of v, $P_v^{'}$, is valid, since $P_u^{max} \leq P_v^{max}$ and $P_v^{'} = P_u \leq P_u^{max} \leq P_v^{max}$; (ii) the new configuration is still valid, as $P_v^{'} = P_u$, and the new configuration is still harmonic. We can repeat the above process until we construct a new configuration $\Theta_{opt}^{'}(P_{i,j})$ with the non-decreasing property held, and with a lower network utilization $U^{'}$. This contradicts with the assumption that $\Theta_{opt}(P_{i,j})$ is the optimal partial configuration with minimum utilization. □

Lemma 2. *Suppose for all* $1 \leq i \leq n$, $P_i^{min} \leq P_{i,j} \leq P_i^{max}$. *The utilization of an optimal valid partial configuration* $\Theta(P_{i,j})$ *is determined by Eq. 4.1.*

$$U(\Theta(P_{i,j})) = \begin{cases} \frac{c_i}{P_{i,j}} & \text{if } i = 1 \\ \frac{c_i}{P_{i,j}} + \min\limits_{y \in \{x | (P_{i-1,x} | P_{i,j})\}} (U(\Theta(P_{i-1,y}))) & \\ & \text{if } 2 \leq i \leq n \end{cases} \tag{4.1}$$

Proof. We prove this lemma by induction. Clearly, $i = 1$ is true because any $P_{1,j}$ forms a harmonic chain itself, and the utilization of the harmonic chain is equal to the utilization of the link. Assuming for $i = k$, the utilization determined by Eq. 4.1 is true. Because of the non-decreasing property of Lemma 1, to derive $U(\Theta(P_{k+1,j}))$, we only need to consider the optimal valid partial configuration $\Theta(P_{k,y})$ where $P_{k,y} | P_{k+1,j}$. Apparently, selecting the minimal $U(\Theta(P_{k,y}))$ minimizes $U(\Theta(P_{k+1,j}))$, which indicates $U(\Theta(P_{k+1,j}))$ is optimal. By mathematical induction the statement of Eq. 4.1 holds for $1 \leq i \leq n$. □

Based on Lemma 2, it is clear that the utilization of the optimal valid configuration for $L = \{L_i\}_{i=1}^{n}$ is the minimal of $U(\Theta(P_{n,j}))$, where $P_n^{min} \leq P_{n,j} \leq P_n^{max}$. Because of the recursive substructure of the period selection problem, we propose a dynamic programming algorithm as shown in Alg. 1.

In Alg. 1, we use an array p to represent the admissible periods for the links. $p[i][j]$ represents the j^{th} admissible period of link L_i within $[P_i^{min}, P_i^{max}]$. c_i is the number of transmission time slots for link L_i. Array u is used to memorize the utilization of the optimal valid partial configurations, and for every optimal valid partial configuration we store the previous period in the harmonic chain in array *prev*. We first initialize all arrays from line 1 to 10. Line 11 to 16 are used to derive the utilization of the optimal valid partial configurations according to Lemma 2. Line 14 ensures that the derived periods form a harmonic chain. After computing the utilization of all optimal valid partial configurations, line 17 selects the minimum utilization for the link set. If a valid optimal configuration is found, the periods for all the links can be retrieved from array *prev* as shown in line 18 to 22. The time complexity of Alg. 1 is $O(n \cdot k^2)$, where $k = \max_{1 \leq i \leq n}(P_i^{max} - P_i^{min})$. It is because for every period within $[P_i^{min}, P_i^{max}]$, it takes at most $O(k)$ time to decide the previous period. For every L_i it takes at most $O(k^2)$ time to decide P_i. Thus, Alg. 1 takes $O(n \cdot k^2)$ time to select periods for all links.

Algorithm 2: Phasing Assignment Algorithm

Input: $s[]$: selected periods from Alg. 1
$c[]$: the corresponding number of transmission time slots of each link
Output: $o[][]$: phasing of each link

```
1  n ← s.length
2  Initialize an array sched of size s[n] with initial value 0
3  for i ← 1 to n do
4      for j ← 1 to c[i] do
5          k ← the smallest index of an element in sched with value 0
6          o[i][j] ← k
7          while k ≤ s[n] do
8              sched[k] ← i
9              k ← k + s[i]
10 return o
```

4.3.2.2 Phasing Assignment

The period selection algorithm finds the optimal valid configuration for the given link set. Based on the selected periods in the period selection stage, we further present a phasing assignment algorithm to assign the proper phasings for the links. We will prove the correctness of the proposed algorithm in Theorem 2.

Alg. 2 presents the structure of the phasing assignment algorithm. The links periods form a harmonic chain, and by Lemma 1 the periods are non-decreasing, so the hyperperiod of the links is the largest period among all the links ($s[n]$). Therefore, in line 2, we create an array *sched* with size $s[n]$ to store the schedule. From line 5 to 6, we assign the first unallocated phasing in the schedule to a fragment of a link. We then fill the schedule for every

instance of the link, as shown in line 7 to 9. For each link, this algorithm takes $O(s[n])$ time to find an empty slot and to fill the schedule. Therefore, to assign phasing for all links, the algorithm takes $O(n \cdot s[n])$ time.

Theorem 2. *Given a link set L in which the periods of all links form a harmonic chain, the utilization bound of L for the JFS problem is* 1.

Proof. We prove this theorem by construction. We claim that for a link set L in which the periods of all the links form a harmonic chain, Alg. 2 can always schedule the link set if the utilization of the link set is ≤ 1. To show the correctness of Alg. 2, we shall show that when we schedule L_k, there should be at least c_k empty time slots for every P_k. Because we schedule the time slots of a link in the first available slots, and the periods of the links form a harmonic chain, the remaining time slots R_k in every P_k are shown in Eq. 4.2.

$$R_k = P_k - \sum_{i=1}^{k-1} c_i \cdot \frac{P_k}{P_i} = P_k \cdot (1 - \sum_{i=1}^{k-1} \frac{c_i}{P_k}) \geq P_k \cdot \frac{c_k}{P_k} = c_k \tag{4.2}$$

Therefore, we can schedule every link L_k in the link set if the utilization is ≤ 1, which means the utilization bound for the link set is 1. □

4.4 Dynamic Link Schedule Adjustment in RT-WiFi Networks

Static data link layer schedule assignment algorithm works well when the network is static and the network topology and link characteristics are known beforehand. In most wireless cyber-physical applications, the network

is dynamic and devices may join and leave the network frequently. In order to support high-speed real-time wireless cyber-physical applications, an on-line scheduling algorithm should be proposed to support dynamic link schedule adjustment and meet the following three design guidelines: First, computational overhead of the on-line algorithm should be minimized to handle communication requests efficiently. Additionally, the on-line algorithm should avoid adjusting existing schedule unless it is necessary. It is because adjusting an existing communication schedule of a communication link not only incurs additional communication overhead for sending the configuration messages but also requires extra control overhead to adjust the sensors and actuators, which results in temporal system instability. Moreover, if changing existing schedule is necessary, the on-line scheduler should carefully adjust the new schedule assignment such that the incurred jitter of the link adjustment is minimized.

To meet the aforementioned design guides, we first provide an overview of our on-line link scheduler. When a new device (link L_i) joins the network, the link scheduler will follow the two stages in the static schedule assignment algorithm to assign the link a proper period and phasing. To minimize the schedule adjustment on existing communication links, the link scheduler will first try to assign a period to the new link without modifying the configurations of existing links. According to Theorem 2, as long as the periods in the new link set form a harmonic chain and the overall network utilization is no larger than 1, the new link set is schedulable and we can assign a proper phasing for the new link. If such a period for the new link cannot be found, we need to

adjust the periods for some of the links in the network. If the periods of the new link set after the adjustment can form a harmonic chain and the network utilization is no larger than 1, the new link will be admitted in the system and allocated with corresponding communication bandwidth. Otherwise, it will be rejected. Next, we will provide detailed operations of our on-line link scheduler in the following subsections.

4.4.1 An Efficient Data Structure for Schedule Management

To handle network dynamics, an efficient link schedule assignment algorithm should be carefully designed to admit link on-line and minimize schedule adjustment overhead in RT-WiFi networks. As described in our communication model, the network manager maintains a superframe which records all the communication links in the network, and its size is the hyperperiod of the current link set. A straightforward implementation may incur significant overhead to handle a join request. For example, to schedule a transmission slot for a new link L_i with period P_i, the network manager needs to first find an empty time slot to derive the phasing ϕ_i for that transmission of L_i, and then check through the entire superframe to make sure that an empty slot always exists in every P_i with the same phasing. This search could be inefficient when the hyperperiod is large. Even worse, if the network manager cannot find a feasible phasing for the new link, it has to adjust the schedule of existing devices. Therefore, to provide efficient schedule construction and adjustment, an effective data structure should be designed for managing the communication

schedule.

Motivated by the mathematical property of a harmonic chain, we designed a tree structure named S-tree to represent the communication schedule in RT-WiFi networks. We assume that there are k periods $P = \{P_i\}_{i=1}^{k}$ for the existing links in the harmonic chain set, and the periods are sorted in the non-decreasing order. We add a new period $P_0 = 1$ as the base of the harmonic chain, and it serves as the root of the S-tree. The root has a depth of zero, and the depth of other nodes in the S-tree is defined as the number of hops from the nodes to the root. In the top-down manner, each level of the S-tree represents a period in the harmonic chain in the non-decreasing order. The degree of a tree node is defined as the number of its children. For example, the degree of the S-tree at level i $(0 \leq i \leq k-1)$ is $\frac{P_{i+1}}{P_i}$, and a leaf node has a degree of zero. Figure 4.1 shows a S-tree with $P = \{1, 2, 4, 8\}$ and its corresponding schedule.

In the S-tree, each tree node represents a subset of time slots in the superframe. The root represents the entire superframe because its period is 1. The child nodes of a tree node collectively represent all the time slots served by their parent. For example, in Figure 4.1, there are four tree nodes at the $P = 4$ level. These four tree nodes represent the four partial schedules in the superframe for the links with a period of 4. To specify which partial schedule the tree node represents, each tree node is assigned with a phasing. The root has a phasing of zero, and we derive the phasings for other tree nodes recursively. Suppose a parent node has a period P, phasing ϕ, and degree

D, the phasings of its children nodes from left to right will be assigned as $\phi + 0 \cdot P, \phi + 1 \cdot P, \cdots, \phi + (D-1) \cdot P$. After the phasing assignment, we can uniquely identify a tree node by the pair $N(P, \phi)$. For example, in Figure 4.1, the phasings at the $P = 8$ level are $\{0, 4, 2, 6, 1, 5, 3, 7\}$, which represent eight partial schedule with period of 8. $N(8, 5)$ and $N(4, 0)$ represent the link schedule of two periodic links A and link B respectively. Maintaining S-tree enables efficient mapping from a tree node to its represented communication schedule.

We define three node status in a S-tree: free, occupied and semi-occupied. A node is marked as **free** if all the time slots represented by the node are available for allocation. A free node is associated with a number, which represent supported period by this node. If the partial schedule represented by a tree node is allocated, then we mark that node as **occupied**. A tree node is marked as **semi-occupied** if some but not all of its descendants are occupied. A semi-occupied node is associated with a set of numbers, which is the union of all the supported periods of its children. Using S-tree can significantly improve the efficiency of the dynamic link scheduling in RT-WiFi networks. For example, when a new link with $P = 4$ and $c = 1$ joins the network, instead of testing every phasing for $P = 4$, the network manager can traverse S-tree from root and find that $N(4, 2)$, and $N(4, 3)$ can support this new link. To validate the efficiency of S-tree, we shall show the time and space complexity of S-tree operations in the next subsection.

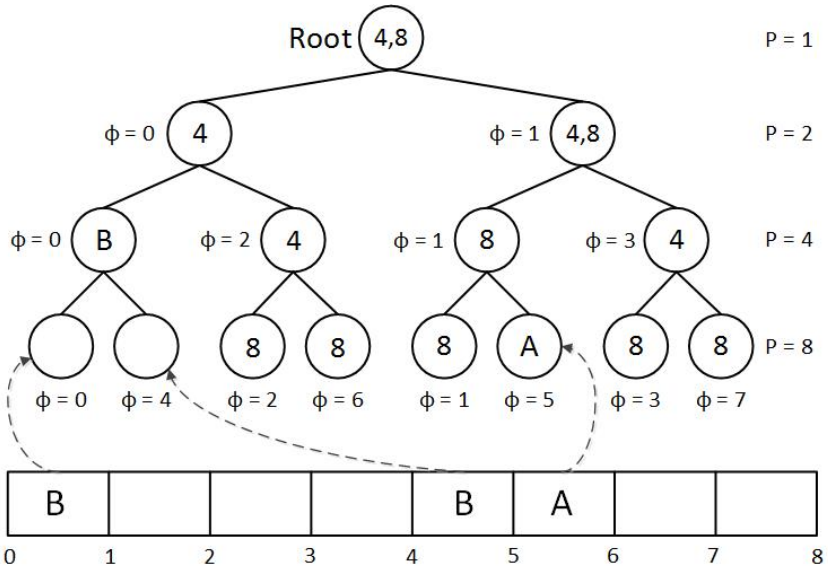


Figure 4.1: Tree representations of schedules.

4.4.2 *S*-tree Operations

S-tree has three operations to support dynamic network management: *Construct*, *Add-link* and *Remove-link*. To discuss the time and space complexity for these operations, we assume that the hyperperiod of the link set is M, and the minimum and maximum degrees of *S*-tree are D_{min} and D_{max}.

4.4.2.1 Construct

The *Construct* operation accepts a harmonic chain of periods as the input parameter and builds a new *S*-tree. The levels of the new *S*-tree represent the number of different periods in the harmonic chain. We mark all the nodes in the *S*-tree as free, and mark the period of this node as the period of this level. Because the link periods form a harmonic chain, the largest period

in the harmonic chain is the hyperperiod of this link set. Since $D_{min} \geq 2$, there are at most $(2M-1)$ nodes in this tree, and the height of S-tree is at most $\log_{D_{min}}(2M-1)$. Free and occupied nodes require $O(1)$ space to store their status. In the worst case, a semi-occupied node requires $O(\log M)$ space to store all the available periods for its children since the height of S-tree is in $O(\log M)$. Therefore, in the worst case, the total space requirement of $(2M-1)$ nodes is $O(M \log M)$. For constructing a S-tree, we require $O(1)$ time to create a node, connect an edge from the node to its parent, and mark up the period for that node. Therefore, a *construct* operation takes $O(M)$ time in total.

4.4.2.2 Add-link

This operation adds one fragment of a link with period P_i to S-tree. For a link with multiple fragments, we repeat this operation for c_i times. There are two stages in the *add-link* operation. The first stage (search stage) is to find a free node by traversing the S-tree from the root. During this traversal, we select a child node with available period $\leq P_i$. If there are more than one node that support period P_i, we traverse any one of them arbitrarily. This stage finishes when we reach a node with period P_i. For example, to add a time slot with period 8 in Fig. 4.1, a valid traversal could be $N(1,0), N(2,1), N(4,1), N(8,1)$. The second stage (update stage) is to update the S-tree information from the assigned node to the root. For example, if node $N(4,3)$ is assigned to a new link, we need to update the status of node

$N(4,3)$, $N(2,1)$, and the root. We mark node $N(4,3)$ as occupied, and change the available period of node $N(2,1)$ to 8. The time complexity of the *add-link* operation is $O(D_{max} \log M)$. It is because in the worst case, both stages need to traverse all tree levels, and in each level we need to look up all children either to decide the next traversal node in the search stage or the available periods of the node in the update stage.

4.4.2.3 Remove-link

The *Remove-link* operation removes one fragment of a link with period P_i from the S-tree. To keep consistency of available periods in the S, this operation updates status of nodes from the removed node up to the root. For example, if $N(8,5)$ in Fig. 4.1 is removed, this operation marks $N(8,5)$, $N(4,1)$, and $N(2,1)$ as free node, and updates the available periods of the root as $(2,4,8)$. The time complexity of this operation is $O(D_{max} \log M)$ because in the worst case it requires to traverse from the lowest level of the S-tree to the root and in each level this operation needs to check at most D_{max} children.

4.4.3 Schedule Management based on S-tree

In this subsection, we present three optimization policies, to further reduce the possible schedule adjustment and the adjustment jitter of communication links.

4.4.3.1 Assignment policy

In add-link operation, if there are more than one node that support the admitted link, we always select the node whose period is the closest to the period of the link. This policy is an assistive policy for *add-link* operation. By preventing low frequency links from unnecessarily occupying nodes that could support high frequency links, this policy aims to reduce future link adjustments when admitting high frequency links. *Assignment policy* is an assistive policy for *add-link* operation. This policy always selects a branch which has the closest period that is larger or equal to the period of the admitted link. For example, in Fig. 4.1, when inserting a single fragment link L_i with $P_i = 8$, *assignment policy* prefers branch $N(2,1)$ to $N(2,0)$ because the available periods of $N(2,1)$ is equal to 8. Similarly, *assignment policy* prefers $N(4,1)$ to $N(4,3)$, and inserts L_i in $N(8,1)$. Later on if a link L_j with $P_j = 4$ and $c_j = 2$ is admitted, we can directly allocate $N(4,2)$ and $N(4,3)$ for P_j without incurring any additional adjustment. *Assignment policy* increases time complexity of *add-link* operation to $O(D_{max}(\log M)^2)$, because deciding the closer branch to the period of the new admitted link between two branches requires $O(\log M)$ time.

4.4.3.2 Replacement policy

If only semi-occupied nodes are available for the admitted link, the semi-occupied node with least number of occupied nodes will be replaced. This policy is a heuristic to reduce the link adjustments when we have to

adjust the schedule for some links in order to add a new link. Suppose a new link with period P_i is admitted, if all available periods in the root are all greater than the requested period, and only semi-occupied nodes are available in level P_i. In this case we traverse all the semi-occupied nodes in level P_i, and select a replacement node according to the *replacement policy*. After that, we adjust all the occupied child nodes of the replaced node by performing several *add-link* operations. This operation takes $O(P_i)+O(\frac{M}{P_i})$ time, since traversing all nodes in level P_i takes at most $O(P_i)$ time, and the replaced semi-occupied node has at most $O(\frac{M}{P_i})$ child nodes. We have noticed a work [37] which may help to further optimize *replacement policy*, however, [37] only considers a binary tree, and it does not consider jitter minimization for node adjustments.

4.4.3.3 Adjustment policy

When adjusting the schedule of a link, we assign a node that has the closest phasing to its original phasing. This policy is an assistive policy for reducing jitters from node adjustments. In the worst case, this operation takes $O(P_i)$ time to traverse all the free nodes in level P_i. Notice that if there are more than one adjusted node in the same level, further optimization could be achieved by using maximum weighted bipartite matching [80]. However, considering n adjusted nodes, this optimization takes $O(n^4)$ which is too costly for an on-line algorithm. Therefore, our scheduler adjusts the schedule of the links one by one.

4.5 Performance Evaluation

We have implemented the proposed harmonic chain based jitter-free data link layer scheduler and the RT-WiFi network manager for efficient device management and network resource allocation. In order to demonstrate our system performance in large-scale network, we conduct a series of simulations to evaluate the proposed algorithms. This section summarizes the important results from our simulation studies on the performance of the proposed algorithms in this chapter.

4.5.1 Simulation Model and Parameter Settings

Our experiments are based on random join and leave requests from RT-WiFi devices to simulate network dynamics. For each simulation, we randomly generate 500 requests. The time interval between two consecutive requests are randomly selected from $[10, d]$, in the unit of time slot, where d controls how frequently a request occurs. We present the results with $d = 200000$ in this section, since we observe similar results when d is varied from 5000 to 1000000. The results are similar when we vary d from 5000 to 1000000, so we only present the results when $d = 200000$ in this section. For an RT-WiFi network with $5kHz$ sampling rate ($200\mu s$ time slot size), a setting with $d = 200000$ represents that the average time interval between two consecutive requests is 20 seconds.

In order to evaluate the performance of the scheduling algorithms under different network utilizations, we use parameter n to represent the expected

number of active nodes in the system. In our experiment, parameter n is configured from 20 to 100 with an increment of 10. For each configuration, we conduct 100 experiments and present the averaged result.

The communication requirement of a join request is controlled by a period range $[P^{min}, P^{max}]$, and the number of fragments c. In order to simulate typical wireless cyber-physical applications, we select link periods according to hardware capabilities and practical communication requirements. For each join request, P^{min} is randomly generated with uniform distribution from $[2, 20]$, which corresponds to setting the maximum supported sampling rate from $250Hz$ to $2500Hz$. This range is set to simulate the maximum supported sampling rates of embedded processors, which typical range from a few hundred hertz to a few thousand hertz. On the other hand, P^{max} is randomly generated from $[10, 600]$ with uniform distribution, corresponding to selecting the minimum required sampling rate of the applications from $8.33Hz$ to $500Hz$. This setting is used to simulate the minimum sampling rate requirements of various sensing and control applications. For example, temperature and blood pressure sensing applications typically require a minimum sampling rate below $10Hz$; for human motion monitoring, the sampling rate is around $50Hz$; for mechanical control, the minimum required sampling rate should be at least $500Hz$. We generate the number of fragments by randomly selecting an integer from the range $[1, 3]$ according to Poisson distribution with $\lambda = 1$. We make this practical selection because most of the sensing/control data can be transmitted in one time slot. According to [77], a $200\mu s$ time slot is able

to transmit 500 bytes payload, which is sufficient for most sensing/control applications.

To compare the performance with our proposed HCJF scheduler, we implement earliest deadline first scheduler (EDF), rate-monotonic scheduler (RM), and the contention-free periodic message scheduler (CF) proposed in [20]. For EDF and RM, P^{max} is used as the period for the link. For CF scheduler, it selects the period of each link to be $2^{\lfloor log_2 P^{max} \rfloor}$. For HCJF scheduler, we apply all the optimization policies designed for S-tree operations.

We evaluate the performance in terms of the following metrics: (1) Average overall jitter: the sum of the jitters of all the admitted links divided by the number of admitted links; (2) Number of adjustment: the total number of schedule adjustment made by all the communication links in a simulation.

4.5.2 Simulation Results

We report the overall jitter with different algorithms in Fig. 4.2. As shown in Fig. 4.2, the average overall jitter of RM and EDF is over 100 times larger than that of HCJF, and 10 to 300 times larger than that of CF. Because the link periods of RM and EDF do not form a harmonic chain, the transmission of lower priority links may be blocked by higher priority links for uncertain amount of time. For this reason, the inter-completion time of two consecutive transmissions under RM and EDF varies, which leads to large jitters. On the other hand, CF and HCJF use period selection to ensure that the periods of all links form a harmonic chain. Due to the property of harmonic chain, if there

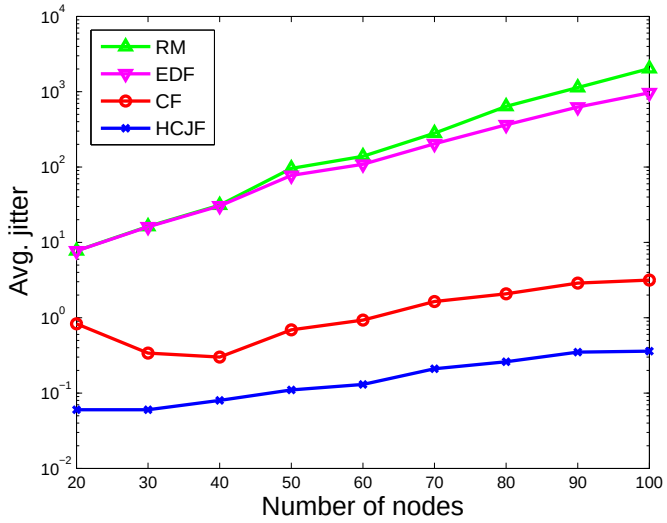


Figure 4.2: Overall Jitter

is no network dynamics, a lower priority link is always blocked by the same set of higher priority links for the same amount of time. Therefore the communication jitters under CF and HCJF schedulers are significantly reduced.

As shown in Fig. 4.2, HCJF has the least overall jitter. Comparing the performance of HCJF with CF, the overall jitter of HCJF is about 85% less in average as shown in Fig. 4.2. HCJF outperforms CF because CF does not consider the future join requests when assigning phasings to the links. Thus CF needs to adjust the schedule of existing links frequently in order to admit new links, which increases jitter to the adjusted links. On the other hand, HCJF is designed to minimize necessary adjustment on existing links when a new link joins. Therefore, network dynamics has minimum impact on the existing

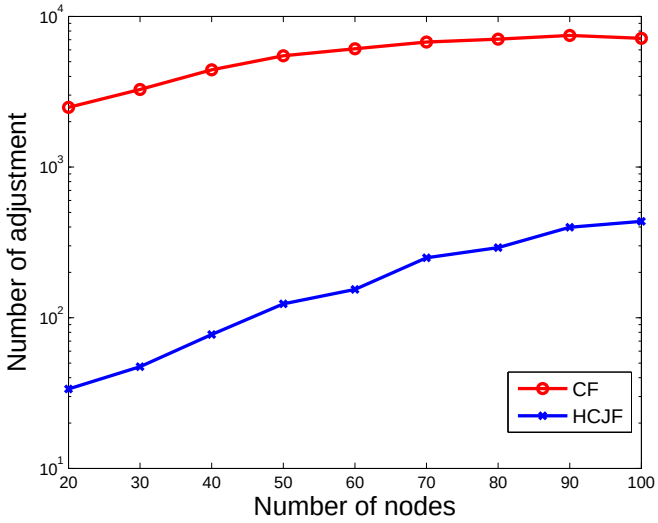


Figure 4.3: Number of Adjustment

links under HCJF. Fig. 4.3 shows the number of adjustments of CF and HCJF in the simulation. From the figure, we observe that HCJF can avoid 94% to 99% of the schedule adjustments in CF. We do not evaluate the performance of RM and EDF in this metric, because EDF and RM are not designed for dynamic network schedule. With present of network dynamic, RM and EDF keep computing the schedule on the fly which may incur considerably large amount of management overheads.

4.5.3 Network Utilization

To evaluate the performance of period selection algorithm in CF and HCJF, we randomly generate static link sets, and compare their total utilization of the link sets. Similar parameter setting as in Section 4.5.1 is used to

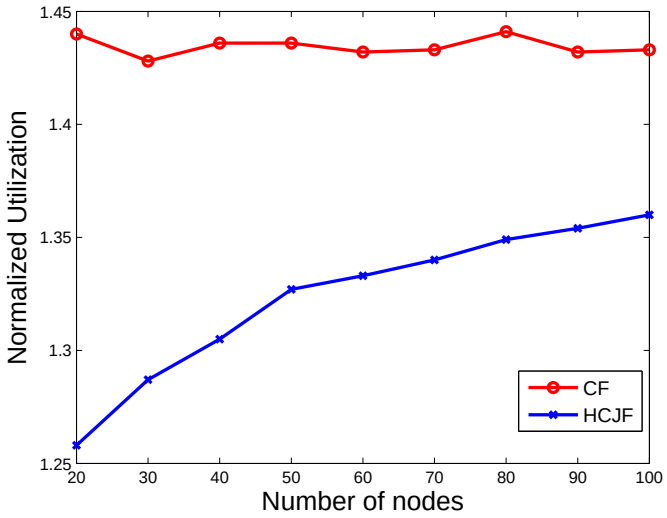


Figure 4.4: Normalized Utilization

generate link periods and communication time. We increase the number of links in a link set from 20 to 100 in a step of 10 to control the total workload. For each configuration, we conduct 100 experiments and present the averaged result. We normalize the collected network utilization to the utilization when every link's period is set to its P^{max}. The normalized utilization represents the increased utilization from a period selection algorithm with respect to the minimum utilization.

We report the normalized utilization of CF and HCJF in Fig. 4.4. As shown in Fig. 4.4, the normalized utilization of HCJF is always smaller than that of CF. Comparing to CF, when number of link is 20 and 100, HCJF reduces 18% and 7% normalized utilization respectively. The normalized utilization of HCJF is lower than CF because CF always selects the period to a

power of two number, which may increase the utilization. On the contrast, HCJF can explore more link periods to form a harmonic chain which leads to lower utilization.

4.5.4 Computational Overhead

To evaluate the computational overhead of HCJF scheduler, we measure the computation time for processing a join request in the RT-WiFi network manager. The network manager runs on a desktop with Intel Core i7-2600 CPU with Ubuntu 14.04 operating system. There are two cases in processing a join request. **Case 1**: a join request can be admitted by fitting its period into the current harmonic chain, and we will assign its phasing according to the dynamic link scheduling as described in Section 4.4. **Case 2**: we have to adjust the period of existing links using Alg. 1, and then assign its phasing as in Case 1. We use a similar experiment setting as described in 4.5.1.

To evaluate the efficiency of our proposed algorithms in a representative worst case, we configured a network with 100 expected active nodes, and present the measurement data in the following. In this experiment, 97.85% of join requests are in Join-Case1. The processing time for Join-Case1 represents the computational overhead of our S-tree operations. The average and maximum processing time for Join-Case1 is $0.01ms$ and $0.26ms$ respectively. This validates that S-tree operations only incur minimal computational overhead. On the other hand, the processing time for Join-Case2 represents the computational overhead of the period selection and reconstruct S-tree. The average

processing time and maximum processing time for Join-Case2 is 9.37ms and 20.11ms respectively. The relatively large computational overhead of period selection is due to the complexity of the period selection algorithm. However, our algorithm does not perform period selection frequently. Only 2.15% of join requests fall into the category of Join-Case2 even when the network is almost fully utilized with 100 expected active nodes. The maximum processing time results from not only the worse case execution time of the period selection algorithm, but also the scheduling delay from operating system, as the network manager is running on a general Ubuntu 14.04 operating system with no real-time enhancement.

Chapter 5

Enhancing Reliability in RT-WiFi Network

In chapter 3, we proposed the RT-WiFi MAC protocol that provides real-time high-speed predictable data delivery and enables designs to meet time-critical industrial needs. However, without explicit reliability enforcement mechanisms, our previous RT-WiFi design is either subject to uncontrolled packet loss due to noise and other interferences, or may suffer from inefficient communication channel usage. In this chapter, we explicitly consider interference from both Wi-Fi and non-Wi-Fi based interference sources, and propose two sets of effective solutions for RT-WiFi network management design. We describe the background information in Section 5.1, and present the system model in Section 5.2. Section 5.3 formalizes the reliable link scheduling problem, and presents a three-stage solution for general interference source. Section 5.4 further considers the interference from regular Wi-Fi network. Finally, we summarize the experimental results in Section 5.5.

This chapter is previously published in [79]. I contributed algorithm design and system implementation to this published work.

5.1 Introduction

Although RT-WiFi can support predictable data delivery, how to provide reliable communication in noisy and harsh environments with stringent timing requirement is far from a completely resolved issue. The reliable transmission mechanisms proposed in Section 3.2 make pessimistic assumptions on the interference traffic, and the link scheduler proposed in Section 4.3 does not consider the communication characteristics for different data links. This can lead to inefficient channel usage, and unsatisfied system performance. On the other hand, most existing work in the literatures either provides bounded-time data delivery under simplified reliability constraints or support reliable communication with limited real-time performance guarantees.

In this chapter, by explicitly considering interference from Wi-Fi and non-Wi-Fi based interference sources, we enhance the RT-WiFi network manager to adapt to the communication schedule dynamically in order to support reliable real-time data delivery and avoid packet loss due to signal attenuation, fading, noise, and interferences from other transmitters. By exploiting available mechanisms in the IEEE 802.11 [3] standard, such as rate adaptation, retransmission, and virtual carrier sensing manipulation, we allow the designer to tailor the wireless system to meet application-specific real-time constraints on the communication links and also meet application-specific reliability requirements. The design highlight of the proposed reliability mechanisms in this chapter is summarized as follows:

- We formalize the real-time reliable link scheduling problem, and propose a real-time rate adaptation algorithm to drive optimal retry chain with minimal transmission time that meets the expected reliability requirement.

- We present an efficient run-time communication link scheduler that has low network management overhead. To utilize channels more effectively, a novel technique for overbooking is invented that improves schedulability of link scheduler significantly.

- To co-exist with regular Wi-Fi networks, we propose a virtual carrier-sensing-based mechanism that not only protects real-time data delivery in the RT-WiFi network but also shares unused bandwidth with regular Wi-Fi networks.

5.2 System Model

In this chapter, we consider the basic single-cluster RT-WiFi network that consists of one RT-WiFi AP and multiple RT-WiFi stations. The RT-WiFi AP and stations form a star network topology. The RT-WiFi network is a TDMA-based time-slotted system, and the basic time unit is a *mini time slot*, which is the minimal indivisible time unit in the communication system. In this system, we consider two reliable communication mechanisms, retransmission and rate adaptation. To improve reliability, if a packet transmission has failed, we can apply retransmission to retransmit the lost packet. There are multiple

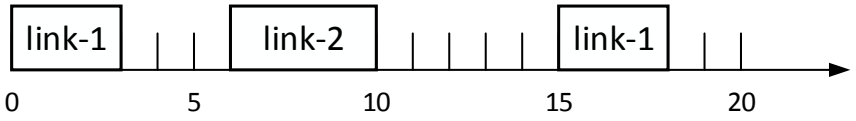


Figure 5.1: An example of a communication link schedule.

data rates available in the RT-WiFi network. By adapting the data rate properly, we can change the underlying physical layer modulation to make it less error-prone to the interference source. However, applying retransmission and adapting to a lower data rate also increase transmission time. Therefore, the network manager needs to schedule the retransmission and adapt data rate carefully to provide real-time reliable communication.

The basic transmission unit for allocating the communical channel in an RT-WiFi communication schedule is a time slot, and a communication link describes the communication activity on a time slot. A communication link specifies the sender, receiver, and the timing information of the time slot. Depending on the application requirements and channel condition of a communication link, the length of a time slot varies. For example, Fig. 5.1 shows an example of a communication schedule with two links. The length of link-1 and link-2 are 2 and 3 mini time slots respectively. We classify links into two types. A downlink is a link with the AP as the sender. It is an uplink if a station is the sender.

The communication schedule of the RT-WiFi network is maintained by the network manager, and is distributed by the AP periodically. The communication schedule is represented by a data structure called the *superframe* which

is a finite sequence of links that can be repeated infinitely to yield an infinite schedule at run time. To adapt to network dynamics, the network manager updates the superframe to provide reliable communication. For every beacon period, the AP compiles the communication schedule with other network management information (i.e. timing information) into a beacon frame, and broadcast it to the RT-WiFi network. Depending on the channel condition, the AP broadcasts the beacon frame multiple times to ensure reliable delivery.

We specify wireless communication by a set of communication links $L = \{L_1, L_2, ..., L_n\}$ where each link L_i represents a communication task that transmits sensing/actuating data periodically. A communication link L_i is associated with a triple of positive integers (P_i, D_i, B_i). P_i and D_i are in the unit of mini time slot, and they denote the period and deadline of a link L_i respectively. We assume $P_i = D_i$ for simplicity. For a real-time communication link, a packet is successfully delivered if it is received by the end of its deadline. B_i is used to represent the number of bytes of the data link layer packet size of link L_i. In our model, we assume the periods of the links form a harmonic chain. That is $\forall 1 \leq i, j \leq n, (P_i \mid P_j) \vee (P_j \mid P_i)$. We make this assumption because of practical system design considerations. In a real system, an application may allow a range of sampling periods, and the harmonic periods of the communication tasks can be efficiently selected by period selection algorithms in the literatures [54, 62].

We introduce the following notations to model the transmission properties with different data rates. A set of K data rates are given as $R =$

$\{R_1, R_2, \ldots, R_K\}$, where R_j denotes the number of bytes that the jth data rate can transmit per mini time slot. For each communication link L_i, we use a set of tuple $\{(p_{(i,j)}, c_{(i,j)})\}_{j=1}^{K}$ to represent the transmission properties with the given data rates. $p_{(i,j)}$ is the probability of a successful data delivery of L_i with data rate R_j, and $c_{(i,j)}$ is defined as the number of mini time slot needed to transmit B_i bytes data when using data rate R_j. To accurately model the transmission time of a packet, we introduce O to account the transmission overhead. For each transmission, the overhead is represented as number of mini time slots and it consists of the time of inter frame space and ACK time. We can derive the transmission time $c_{(i,j)}$ of link L_i as $c_{(i,j)} = O + \lceil B_i/R_j \rceil$. We apply ceiling to calculate transmission time because we do not use the empty space left in a mini time slot.

To support reliable communication, we use Q_i to designate the expected packet delivery ratio for link L_i. To achieve the expected packet delivery level, we can schedule a retry chain that consists of a series of data retransmission with the available data rates. A retry chain of L_i with m_i retry is denote as $T_i = \{T_{(i,1)}, T_{(i,2)}, \cdots, T_{(i,m_i)}\}$, where $1 \leq T_{(i,j)} \leq K$ for $1 \leq j \leq m_i$. $T_{(i,j)}$ represents the index of the data rates. For example, a retry chain $T_i = \{2, 3\}$ means that L_i will first transmit with data rate R_2. If the first transmission is failed, it then uses data rate R_3 to retransmit again. We further define the transmission time and delivery rate of a retry chain as follows.

Definition 6. *Given a retry chain $T_i = \{T_{(i,j)}\}_{j=1}^{m_i}$, the transmission time of T_i is defined as $X(T_i) = \sum_{j=1}^{m_i} c_{(i,T_{(i,j)})}$.*

Definition 7. *Given a retry chain* $T_i = \{T_{(i,j)}\}_{j=1}^{m_i}$*, the expected data delivery rate of* T_i *is defined as* $E(T_i) = 1 - \prod_{j=1}^{m_i}(1 - p_{(i,T_{(i,j)})})$.

5.3 Reliable Transmission for General Interference Source

In this section, we discuss how to provide real-time reliable data transmission for general interference source in RT-WiFi network. We take advantage of rate adaptation and retransmission to construct a reliable TDMA communication schedule in order to provide reliable data transmission on a lossy channel. We first formulate the scheduling problem, and present the hardness result of this problem. Due to the hardness of this problem, we adopt a three-stage approach to tackle this problem.

5.3.1 Real-Time Reliable Link Scheduling Problem

Definition 8. *Real-Time Reliable Link Scheduling (RTRLS) Problem: Given a set of data rates* $R = \{R_i\}_{i=1}^{K}$*, a set of communication links* $L = \{L_i\}_{i=1}^{n}$ *with a set of harmonic periods* $P = \{P_i\}_{i=1}^{n}$*, expected packet delivery ratio* $Q = \{Q_i\}_{i=1}^{n}$*, and each link* L_i *is associated with its transmission properties* $\{(p_{(i,j)}, c_{(i,j)})\}_{j=1}^{K}$*. The RTRLS is to determine a retry chain* T_i *and a phasing* ϕ_i *for each link* L_i*, such that* ***(C-1)*** $E(T_i) \geq Q_i$ ***(C-2)*** *the* j*-th instance of link* L_i *is transmitted on* $[\phi_i + j \cdot P_i, \phi_i + j \cdot P_i + X(T_i))$*; and* ***(C-3)*** *only one link is transmitted at a mini time slot.*

In the RTRLS problem, we require the TDMA schedule of the communi-

cation links to be strictly periodic because of practical system implementation considerations. A strictly periodic schedule reduces the overhead of schedule distribution, simplifies interrupt handler design, and could potentially save energy consumption for the network subsystems. We evaluate the schedule distribution overhead under different schedulers in Section 5.5.3.

Theorem 3. *To decide if an instance of the RTRLS problem is schedulable or not is NP-hard.*

Proof. To show that the RTRLS problem is NP-hard, we reduce the 3-partition problem [39] into the RTRLS problem. As shown in [39], the 3-partition problem is NP-Complete in the strong sense. The input of 3-partition problem is as following. Given a finite set A of $3m$ elements, a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$, a bound $B \in \mathbb{Z}^+$. Each $s(a)$ satisfies $\frac{B}{4} < s(a) < \frac{B}{2}$, and $\sum_{a \in A} s(a) = mB$. The 3-partition problem ask if A can be partitioned into m disjoint set $S_1, S_2, \ldots, S_m$ such that $\sum_{a \in S_i} s(a) = B$, where $1 \leq i \leq m$.

We reduce an instance of the 3-partition problem into the RTRLS problem as follows. We create one data rate in the data rate set $R = \{1\}$. We construct link L_1 where $P_1 = (B + 1)$, and transmission condition $p_{(i,1)} = 1.0$ and $c_{(1,1)} = 1$. For each element $a \in A$, we create a communication link L_i where $P_i = m(B+1)$, and transmission condition $p_{(i,1)} = 1.0$ and $c_{(i,1)} = s(a)$. Under this setting, the periods of the communication links form a harmonic chain. For each element in A it takes $O(1)$ time for the reduction. Thus, this reduction can be done in polynomial time.

If there exists a solution in 3-partition problem, we set all the retry chain as $\{T_i = \{1\}\}_{i=1}^{3m+1}$, and $\phi_1 = 0$. Suppose we order the elements in the disjoint set S_i arbitrary, then $S_i = \{s(a)_{(i,1)}, s(a)_{(i,2)}, \cdots, s(a)_{(i,|S_i|)}\}$. For the j-th element in S_i, we set the corresponding phasing as following

$$\begin{cases} (i-1)\cdot(B+1)+1 & j=1 \\ (i-1)\cdot(B+1)+1+\sum_{k=1}^{j-1} s(a)_{i,k} & j>1 \end{cases}$$

This is a valid solution for the RTRLS problem since the expected delivery rate is fulfilled, and only one link is transmitted at a mini time slot.

On the other hand, if there is a solution for the RTRLS problem, we can add the corresponding element of link $\{L_i\}_{i=2}^{3m+1}$ to a disjoint set S_k depending on ϕ_i as following

$$k = \begin{cases} \lfloor \frac{\phi_i - \phi_1}{(B+1)} \rfloor & ,1 \leq \lfloor \frac{\phi_i - \phi_1}{(B+1)} \rfloor \leq m \\ m & ,otherwise \end{cases}$$

It is a valid solution for the 3-partition problem because L_1 is a strictly periodic task, and there are exactly m mini time slots between every instance of L_1. Therefore, there is also a solution for the 3-partition problem. This completes the proof. □

Since the RTRLS problem is NP-hard, we are more interested in finding a practical solution for our typical use case. We propose the following three-stage approach to tackle the RTRLS problem. In Section 5.3.2, we consider condition **C-1** in the RTRLS problem, and present the real-time rate adaptation algorithm that generates a retry chain for each link to fulfill the expected

reliability requirement with minimal transmission time. Given the transmission time of each link, we then consider condition **C-2** and **C-3** of the RTRLS problem, and present a heuristic to construct the TDMA communication link schedule in Section 5.3.3. Finally, we introduce an overbooking technique in Section 5.3.4 to further improve the schedulability of the proposed scheduler.

5.3.2 Real-Time Rate Adaptation

This section presents the real-time rate adaptation algorithm. The goal for the real-time rate adaptation algorithm is to derive a retry chain for a link such that the expected packet delivery rate is fulfilled and the transmission time of the retry chain is minimized. We formalize the rate adaptation problem as follows:

Definition 9. *Real-Time Rate Adaptation (RTRA) Problem: Given K data rates, and a communication link L_i with deadline D_i, expected packet delivery ratio Q_i and transmission properties $\{(p_{(i,j)}, c_{(i,j)})\}_{j=1}^{K}$. The RTRA problem is to determine a retry chain $T_i = \{T_{(i,j)}\}_{j=1}^{m_i}$ such that $E(T_i) \geq Q_i$, $X(T_i) \leq D_i$ and $X(T_i)$ is minimized.*

Lemma 3. *Given a time budget $t \leq D_i$, and we define $OPT_i(t)$ as the minimal*

packet loss rate for L_i. The $OPT_i(t)$ is determined by the following equation.

$$OPT_i(t) = \begin{cases} 1.0 & ,if\ t = 0 \\ min \left\{ \begin{array}{c} OPT_i(t-1) \\ \min\limits_{1 \le j \le K} \{OPT_i(t - c_{(i,j)}) \cdot (1 - p_{(i,j)})\} \end{array} \quad ,c_{(i,j)} \le t \right\} & \\ & ,otherwise \end{cases} \tag{5.1}$$

Proof. We proof this lemma by induction. For $t = 0$, it is clear that the packet loss rate is 1.0. Assuming that for $t = n$, the packet loss rate determined by the Eq. 5.1 is minimal. Then for $OPT_i(n+1)$, there are two cases. For the first case, we simply use the same retry chain as $t = n$. Therefore, the loss rate is same as $OPT_i(n)$. For the second case, if the current time budget $(n+1)$ is greater or equal to the transmission time of data rate R_j, then we schedule R_j at the end of retry chain. The packet is lost if all the retries with time budget $t = (n+1) - c_{(i,j)}$ and the last retry are failed. Thus, the minimal packet loss rate with R_j is $OPT_i(n+1-c_{(i,j)}) \cdot (1 - p_{(i,j)})$. Then we select the minimal loss rate among all the K data rates. Finally, we choose the smaller loss rate among the previous two cases as $OPT_i(n+1)$, which is minimal. By mathematical induction the statement of Eq. 5.1 holds for $1 \le t \le D_i$. □

Based on the recursive substructure in Lemma 3, we propose a dynamic programming based algorithm that derives a retry chain with the minimal time budget. Alg. 3 shows our real-time rate adaptation (RTRA) algorithm. In this algorithm, array $loss[]$ denotes the minimal packet loss rate under a particular time budget, and the array $rates[]$ is used to derive a chain of data rates to

Algorithm 3: Real-Time Rate Adaptation Algorithm

Input: K data rates, and a communication link L_i with deadline D_i, expected packet delivery ratio Q_i and transmission properties $\{(p_{(i,j)}, c_{(i,j)})\}_{j=1}^{K}$.

Output: The retry chain $T_i[]$. It is empty if no valid retry chain is found.

```
1  loss[0] = 1.0
2  for x ← 0 to D_i do
3      rate[x] = −1
4  for x ← 1 to D_i do
5      loss[x] = loss[x − 1]
6      for y ← 1 to K do
7          if x ≥ c_(i,y) and loss[x − c_(i,y)] · (1 − p_(i,y)) ≤ loss[x] then
8              loss[x] = loss[x − c_(i,y)] · (1 − p_(i,y))
9              rate[x] = y
10     if (1 − loss[x]) ≥ Q_i then
11         break
12 if 1 − loss[x] ≥ Q_i then
13     return GetRetryChain(x, rate[], {c_(i,j)}_{j=1}^{K})
14 return null
15 Procedure GetRetryChain(x, rate[], {c_(i,j)}_{j=1}^{K})
16     y = 1
17     while x > 0 do
18         if rate[x] ≠ −1 then
19             T_i[y] = rate[x]
20             x = x − c_(i,rate[i])
21             y = y + 1
22         else
23             x = x − 1
24     return T_i[]
```

achieve the minimal packet loss rate. We first initialize all arrays from line 1 to 3. Line 4 to 11 are used to derive the minimal packet loss rate for time budget from 1 to D_i. As in line 5, the minimal packet loss rate for a time budget x is at least as small as the loss rate with time budget $x-1$. In line 6 to 9, we try all the K data rates to test if we can achieve a lower packet loss rate. We break from the search loop if the expected packet delivery rate is fulfilled in line 11. If the expected data delivery rate could be met, we retrieve the retry chain in array T_i by using procedure GetRetryChain(). The real-time rate adaptation algorithm is a pseudo-polynomial time algorithm. It is because for each time budget it takes $O(K)$ time to calculate the packet loss ratio of a data rate. Thus, time complexity of the RTRA algorithm is $O(D \cdot K)$.

5.3.3 Communication Link Scheduling

In this section, we discuss the second stage for solving the RTRLS problem that discuss how to schedule a set of communication links when the retry chains of the communication links are determined. We focus on assigning phasings to the links such that condition **C-2** and **C-3** in the RTRLS problem are fulfilled. We formally define the communication link scheduling problem as below.

Definition 10. *Communication Link Scheduling (CLS) problem: Given a set of communication links* $L = \{L_i\}_{i=1}^{n}$ *with a set of harmonic periods* $P = \{P_i\}_{i=1}^{n}$ *and a set of retry chains* $T = \{\{T_{(i,j)}\}_{j=1}^{m_i}\}_{i=1}^{n}$. *The CLS problem is to determine a set of phasings* $\phi = \{\phi_i\}_{i=1}^{n}$ *such that (1) the* j*-th instance of link*

L_i is transmitted at $[\phi_i + j \cdot P_i, \phi_i + j \cdot P_i + X(T_i))$; and (2) only one link is transmitted at a mini time slot.

Theorem 4. *To decide if an instance of the CLS problem is schedulable or not is NP-hard.*

Proof. It can be proved similarly as Theorem 3. □

As we discussed in Section 2.3, the CLS problem is similar to scheduling a set of non-preemptive strictly periodic tasks on an uni-processor system. Due to the hardness of the CLS problem, we focus on providing an efficient run-time algorithm for link scheduling with low network management overhead. We present our link scheduling heuristic as shown in Alg. 4.

Alg. 4 presents the structure of the link scheduling algorithm. Since the periods of the links form a harmonic chain, the hyperperiod of the links is the largest period among all the links. From line 1 to 2, we initialize a boolean array $s[]$ of size P_n to indicate if a phasing is assigned. From line 3 to 20, we assign phasings to links in the order of period size. Since the task periods are in a harmonic chain, the schedule of a smaller period repeats multiple time in larger periods. From line 3 to 8, if we detect P_i is larger than P_{i-1}, we repeat the schedule in $[0, P_{i-1} - 1]$ $\frac{P_i}{P_{i-1}}$ times before we schedule link L_i. From line 9 to 20, we loop through P_i and check if there are enough unallocated slots that could fit L_i. If there are enough empty slots that can fit the retry chain T_i, we assign ϕ_i for L_i and mark down the schedule in line 18 to 20. The time complexity of Alg. 4 is $O(n \cdot P_n)$.

Algorithm 4: Shortest Period First Scheduler

Input: A set of communication links $L = \{L_i\}_{i=1}^{n}$ with $P = \{P_i\}_{i=1}^{n}$ and $T = \{\{T_{(i,j)}\}_{j=1}^{m_i}\}_{i=1}^{n}$.

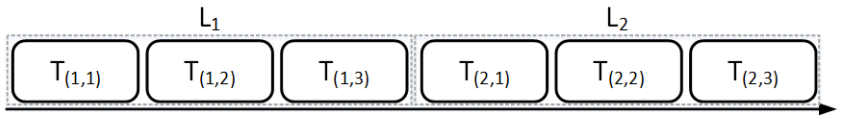
Output: The phasings of the links $\phi = \{\phi_i\}_{i=1}^{n}$. A boolean value is returned to indicate if the links is schedulable.

```
// W.L.O.G. we assume the links are sorted by their
   periods in non-decreasing order.
```

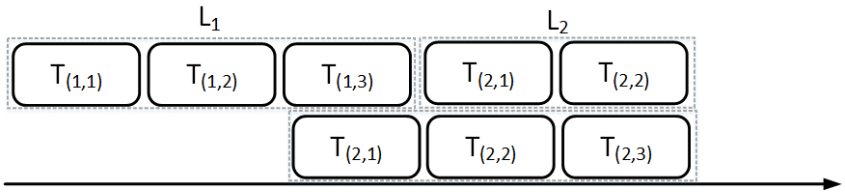
```
1  for i ← 0 to P_n − 1 do
2      s[i] = false
3  for i ← 1 to n do
4      if i ≠ 1 and P_i > P_{i−1} then
5          b = P_i / P_{i−1};
6          for j ← 1 to b − 1 do
7              for k ← 0 to P_{i−1} − 1 do
8                  s[j · P_{i−1} + k] = s[k]
9      j ← 0 k ← 0
10     while k < X(T_i) do
11         if j + k > P_i then
12             return false;
13         if s[j + k] = true then
14             j = j + k + 1;
15             k ← 0
16         else
17             k ← k + 1
18     φ_i = j;
19     for k ← 0 to X(T_i) − 1 do
20         s[j + k] = true
21 return true;
```

5.3.4 Schedulability Enhancement with Overbooking

In this section, we present a technique called overbooking to further enhance the schedulability of the communication link scheduling. Overbooking technique is based on a key observation that the last retry in a retry chain has relatively low chance to be executed. We demonstrate the benefit of overbooking scheduling with a motivating example as shown in Fig. 5.2. There are two links L_1 and L_2 with the same parameter sets as following. $P_1 = P_2 = D_1 = D_2 = 5$ $Q_1 = Q_2 = 0.8$, $p_{(1,1)} = p_{(2,1)} = 0.5$, and $c_{(1,1)} = c_{(2,1)} = 1$. By using the proposed RTRA algorithm, we get $T_1 = T_2 = \{1, 1, 1\}$, and $E(T_1) = E(T_2) = 0.875 \geq Q_1 = Q_2$. Since $X(T_1) = X(T_2) = 3$ and both of their deadline are 5, we can not schedule both links in Fig. 5.2a. However, we observe that the probability for L_1 to use the last retry is only 0.25, since L_1 only uses the third retry if both the first two reties are failed. Therefore, if there is an efficient mechanism that could detect L_1's usage of its last retry, then as shown in Fig. 5.2b, we can overbook mini time slot 3 such that $T_{(2,1)}$ only transmits if $T_{(1,3)}$ does not use the channel in mini time slot 3. Moreover, we need to make sure both L_1 and L_2 still meet their reliability requirements with overbooking mechanism. For L_1, since $T_{(1,3)}$ is always transmitted when it is necessary, the reliability requirement of L_1 is fulfilled. Additionally, since L_2 can only transmitted at mini time slot 3 when $T_{(1,3)}$ is not transmitted, the successful packet delivery rate for L_2 is calculated as $0.75 \cdot E(\{T_{(2,1)}, T_{(2,2)}, T_{(2,3)}\}) + 0.25 \cdot E(\{T_{(2,1)}, T_{(2,2)}\}) = 0.75 \cdot 0.875 + 0.25 \cdot 0.75 = 0.84375 \geq Q_2$. Thus, both L_1 and L_2 could be scheduled with expected re-



(a) Communication schedule without overbooking.



(b) Communication schedule with overbooking.

Figure 5.2: Overbooking example

liability. In the following subsection, we shall present the conflict resolution mechanism and elaborate how to construct a schedule with overbooking mechanism.

5.3.4.1 Efficient Conflict Resolution Mechanism with Overbooking

One way to reuse the spare time of an unused retry time is to let the network manager to send a management message explicitly to notify the overbooked device to transmit. However, this message based conflict resolution mechanism is inefficient since the communication overhead of a notification message could be larger than the short unused spare time. On the other hand, we propose a novel approach that uses the characteristics of the wireless media to detect the unused spare time. We consider two links L_i and L_j, assuming L_i is already scheduled and we want to decide if L_j can be scheduled right after L_i with overbooking. Table 5.1 summarizes the conditions to resolve the

Table 5.1: Overbooking Conditions

L_i	L_j	Overboook?	Reason
Down link	Down link	Yes	AP can only transmit L_j after it finishes transmission of L_i.
Down link	Up link	Yes	Station that transmits L_j can carrier sense AP's transmission of L_i.
Up link	Down link	Yes	AP can only transmit L_j after it finishes receiving L_i.
Up link	Up link	Depends	Only if the source node of L_i and L_j are the same station. Otherwise, the two stations may not carrier sense each other.

conflict in the schedule with overbooking.

5.3.4.2 Overbooking Scheduling

Based on the aforementioned conditions, we can determine if a conflict in the overbooking schedule can be efficiently resolved without additional management overhead. We shall further calculate the transmission time and the expected packet delivery rate of a link L_j if it is overbooked with its previous link L_i. In the following, we assume L_i is already scheduled with $T_i = \{T_{(i,j)}\}_{j=1}^{m_i}$, and we want to schedule L_j with the overbooking technique to satisfy Q_j.

We first introduce Alg. 5 that can derive the retry chain T_i with maximal packet delivery rate for a link L_i with a given time budget t_i. Given a time budget t_i, Alg. 5 obtains a retry chain for a communication link L_i with the maximal expected packet delivery rate. This algorithm is modified from Alg. 3,

Algorithm 5: Real-Time Rate Adaptation Algorithm with a Given Time Budget

Input: A communication link L_i with transmission properties $\{(p_{(i,j)}, c_{(i,j)})\}_{j=1}^{K}$, and a time budget t_i.

Output: An array $T_i[]$ of the retry chain. It is empty is no valid retry chain is found.

```
1  Algorithm MaxRateRetry(L_i, t_i)
2      loss[0] = 1.0
3      for x ← 0 to t_i do
4          rate[x] = −1
5      for x ← 1 to t_i do
6          loss[x] = loss[x − 1]
7          for y ← 1 to K do
8              if x ≥ c_(i,y) and loss[x − c_(i,y)] · (1 − p_(i,y)) ≤ loss[x]
                then
9                  loss[x] = loss[x − c_(i,y)] · (1 − p_(i,y))
10                 rate[x] = y
11     return GetRetryChain(x, rate[], {c_(i,j)}_{j=1}^{K})
```

and it uses the $GetRetryChain()$ procedure in Alg. 3 to derive the retry chain. It has the same time complexity as Alg. 3. We define Alg. 5 as an operation, $MaxRateRetry()$, and we derive a retry chain as $T_i = \text{MaxRateRetry}(L_i, t_i)$.

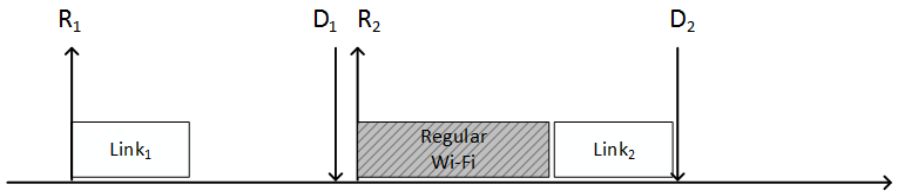
We then define $prob_i$ as the probability if L_i does not access the channel in its last retry. T_j and T'_j represent the retry chain of L_j when L_i does not access the channel in its last retry and when L_i utilizes its last retry, respectively.

$$prob_i = 1 - \prod_{j=1}^{m_i - 1} (1 - p_{(i,T_{(i,j)})}) \tag{5.2}$$

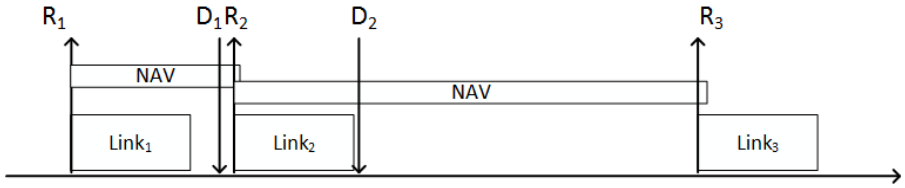
Depending on whether L_i utilizes its last retry , we minimize time budget t_j as follows:

$$\begin{aligned} \text{minimize} \quad & t_j \\ \text{subject to} \quad & T_j = MaxRateRetry(L_j, t_j + c_{(i,m_i)}) \\ & T'_j = MaxRateRetry(L_j, t_j) \\ & prob_i \cdot E(T_j) + (1 - prob_i) \cdot E(T'_j) \geq Q_j \end{aligned} \tag{5.3}$$

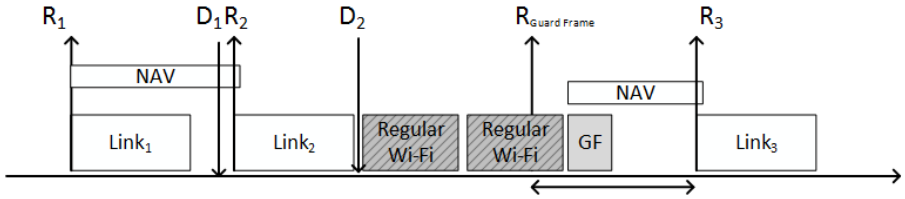
To calculate the minimal time budget t_j, we modify Alg. 5 to do a linear search, and stop when Q_j is met. Therefore, the time complexity with overbooking technique remains the same as the RTRA algorithm. Please note that in the worst case, the overbooking technique fails back to the original RTRA algorithm when $prob_i \to 0$. The overbooking technique effectively reuses the unused channel time and improves schedulability. We shall evaluate its performance in Section 5.5.3.



(a) Original co-existence mode in Section 3.2.3.3.



(b) Co-existence mode with virtual carrier sense.



(c) Co-existence mode with bandwidth sharing.

Figure 5.3: Co-existence mode

5.4 Reliable Transmission with Regular Wi-Fi Networks

In this section, we discuss how to provide reliable data transmission for an RT-WiFi network when it coexists with regular Wi-Fi networks. We assume the nodes in RT-WiFi network and the regular Wi-Fi networks could carrier sense each other. Based on this assumption, we propose mechanisms to prioritize the data transmission in RT-WiFi network and share the remaining bandwidth to the regular Wi-Fi network.

5.4.1 Deferring Regular Wi-Fi Traffic with Virtual Carrier Sensing

Our first proposed mechanism is to utilize virtual carrier sensing to actively deferring the transmission of regular Wi-Fi traffic. In the IEEE 802.11 standard [3], virtual carrier sensing provides an indication of a busy media, and a regular Wi-Fi node defers its media access if it either physically or virtually senses the media is busy. Virtual carrier sensing is implemented by the network allocation vector (NAV) mechanism. When a MAC layer frame is transmitted, it estimates the frame transmission time in micro seconds and specifies the time at the duration filed in the MAC frame header. If a node receives a Wi-Fi MAC layer frame, the node will set its NAV and keep the virtual carrier sensing on till the end of the duration.

We compare the new carrier sensing approach with the original co-existence mode that we proposed in Section 3.2.3.3. As shown in Fig. 5.3a, since the old co-existence mode can not defer the transmission of regular Wi-Fi traffic, it pessimistically estimates the worst case transmission time of the regular Wi-Fi traffic assuming regular Wi-Fi uses the maximum transmission unit size and the lowest data rate. Thus, this approach leads to inefficient channel usage. On the other hand, in Fig. 5.3b, we can schedule $Link_2$ right after the deadline of $Link_1$ with virtual carrier sensing mechanism. The transmission of $Link_2$ will not be blocked by regular Wi-Fi traffic even if $Link_1$ does not use all of its retry chain, since regular Wi-Fi will be deferred till the end of NAV specified by $Link_1$.

5.4.2 Sharing Unused Bandwidth with Regular Wi-Fi Networks

Although virtual carrier sensing can effectively defer the data transmission of regular Wi-Fi networks, however, aggressively use virtual carrier sensing may completely block traffic in regular Wi-Fi networks. For example, in Fig. 5.3b, we use NAV to block a long channel idle time from D_2 to R_3 that wastes the sharing opportunity to regular Wi-Fi traffic. In the following, we present a mechanism to protect RT-WiFi traffic and share unused bandwidth with regular Wi-Fi networks.

Our bandwidth sharing mechanism consists of guard frame injection and the channel usage estimation. The guard frame is a management frame in the RT-WiFi network that broadcasts NAV to defer transmission of traffic from regular Wi-Fi networks. The channel usage estimation scheme logs the past channel access time of regular Wi-Fi traffic by measurement and it estimates the access time by using exponential weighted moving average. Suppose the estimated channel access time for regular Wi-Fi traffic is t_{Est}, and the transmission time of a guard frame is $t_{\text{GuardFrame}}$, we define the sharing threshold t_{Shared} as following.

$$t_{\text{Shared}} = t_{\text{Est}} + t_{\text{GuardFrame}} \tag{5.4}$$

As shown in Fig. 5.3c, if the channel idle time is less than t_{Shared}, we use virtual carrier sensing to protect RT-WiFi traffic. On the other hand, if the channel idle time in our communication schedule exceeds the sharing threshold, we schedule a guard frame and share the used channel time to regular Wi-Fi

traffic. Notice that because we reserve time to tolerate both the guard frame and the channel access from regular Wi-Fi traffic, the transmission of RT-WiFi traffic will not be delayed if the estimation is accurate. However, if the transmission of an RT-WiFi link misses its deadline because we underestimate the channel access time, our system will update the channel access estimation adaptively and construct a new communication schedule.

5.5 Performance Evaluation

We have implemented the proposed algorithms, and conducted a series of experiments to evaluate the system performance. We first describe our experimental setup, and present two sets of simulations to evaluate the performance of real-time rate control algorithm and the communication link scheduler. We then present experiments on our test bed to evaluate the effectiveness of the co-existence mechanisms with regular Wi-Fi network.

5.5.1 System Setup

The following settings and parameters are used in our experiment. We run the simulations in Section 5.5.2 and Section 5.5.3 on a Linux desktop with Intel Core i7-2600 CPU with Ubuntu 14.04 operating system, and we obtain communication parameters from IEEE 802.11 standard [3]. We use IEEE 802.11a as an example, and the parameters may be derived similarly for other physical layer technologies. There are 8 available data rates in IEEE 802.11a, ranging from 6Mbps to 54Mbps. For each retry, the communication overhead

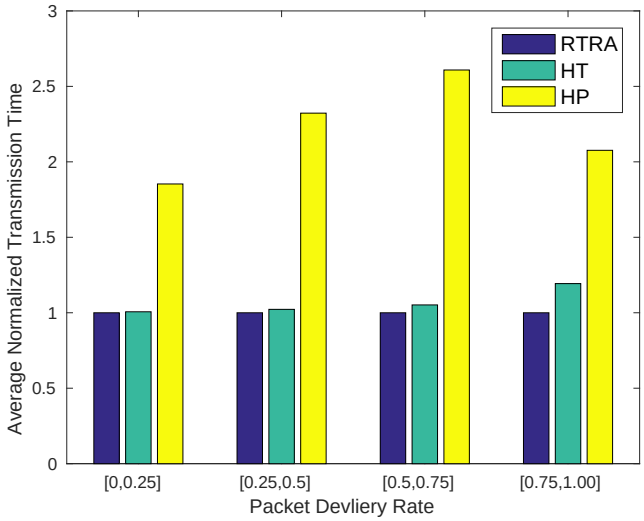


Figure 5.4: Rate Adaptation Performance with Different Channel Condition.

O consists of a PIFS, a SIFS, and an ACK timeout, which is $25+16+25=66$ (μs) in total. In the RT-WiFi protocol, the mini time slot size is set as 100μs, and the expected packet delivery rate for each link is 0.9. For the experiments in Section 5.5.4, we run our test bed in our lab located at UT GDC building 5F. We deploy one RT-WiFi AP on a laptop with Intel Core i3 2320M CPU, and we use six Intel Galileo boards as the stations. Each RT-WiFi node is equipped with Atheros AR9280 wireless card.

5.5.2 Rate Adaptation Algorithm Performance Evaluation

In this subsection, we evaluate the performance of the proposed real-time rate adaptation (RTRA) algorithm in Sec. 5.3.2. For each simulation, we randomly generate 100000 links to simulate the performance of rate adaptation

algorithm under various channel conditions. We compare the RTRA algorithm with Linux kernel's default rate control algorithm minstrel [10]. While minstrel does not aware of deadline of a link, we utilize two heuristics that minstrel uses to construct retry chains and compare the transmission time of the retry chains. The two heuristics, high throughput (HT) and high probability (HP), use the data rate with the highest throughput and highest packet delivery rate to build retry chains respectively. We construct retry chain to meet the expected packet delivery rate for the three methods, and compare the transmission time of the retry chains. Since the RTRA algorithm can always get a retry chain with minimum transmission time, we report the transmission time of HT and HP heuristics normalized to the transmission time of RTRA.

In this experiment, we set the packet size to 1500 bytes which is the maximum transmission unit of the MAC layer, and compare the transmission time of retry chains with different packet delivery rates. We classify the packet delivery rates into four groups. When a link is generated, it randomly picks the packet delivery rate for each data rate within the range of a group. We report the average normalized transmission time in Fig. 5.4. As shown in Fig. 5.4, RTRA and HT always outperform HP because a data rate with high delivery rate does not always lead to short transmission time. On average, RTRA reduces packet transmission time over HT by around 20% when the packet delivery rate is within $[0.75, 1.0]$, and they have similar average performance when the packet delivery rate is low. This is because the retry chain is longer with low packet delivery rates, the retry chain of RTRA is similar to HT in

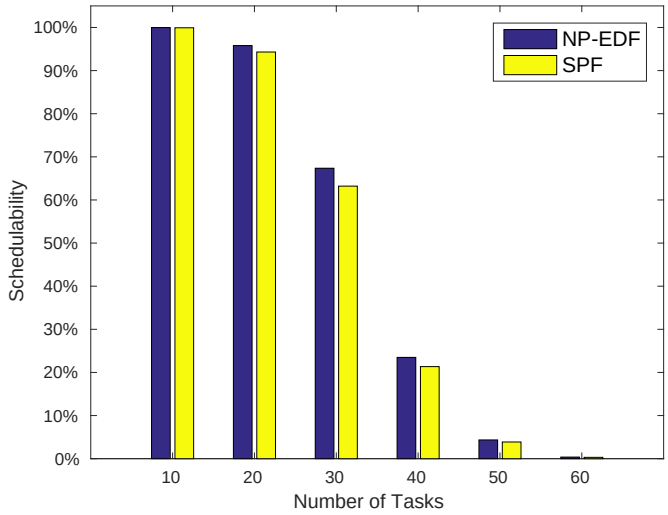


Figure 5.5: Schedulability comparison of NP-EDF and SPF.

that case. However, in the worst case, we observe 10% to 55% transmission time reduction of RTRA over HT.

5.5.3 Performance Evaluation of Link Scheduler

We then evaluate the performance of our shortest period first (SPF) link scheduler. We are interested in the schedulability and network management overhead of proposed link scheduler. We compare SPF scheduler with non-preemptive earliest deadline first (NP-EDF) scheduler. For NP-EDF, we assume all links are ready at time 0, and NP-EDF always schedules a link with earliest deadline. To evaluate schedulability, we randomly generate 100000 task sets and compare the number of schedulable task sets of the two algorithms. We control the number of communication links to control the work-

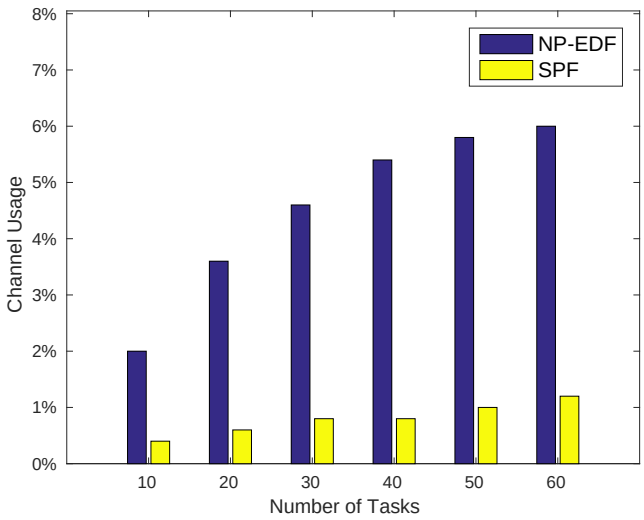


Figure 5.6: Network management overhead Comparison.

load, and we select the parameters for communication links randomly by uniform distribution in the following range. The packet size $B_i \in [100, 1500]$, the packet delivery rate $\in [0.75, 1.00]$, and the periods in the following harmonic chain $= (2.5, 5, 10, 50, 100, 200, 1000)$ ms. This harmonic chain represents sampling rate from 1Hz to 400Hz, which is representative for various type of applications. To compare network management overhead, we use the network setup as described in Section 5.5.1. We assume it takes 6 bytes to specify a time slot in the TDMA schedule, which includes two bytes for specifying a link ID, two bytes for a time slot phasing, and two bytes for the duration of a time slot.

Fig. 5.5 reports the percentage of schedulable task sets for SPF and NP-EDF schedulers. When the number of tasks are between $[10, 20]$, both

schedulers can schedule most of the task sets. NP-EDF outperforms SPF at most by 4%, when there are 30 to 40 tasks in a task set. When the number of tasks are more than 50, because of the high workload, both schedulers can only schedule less than 5% of the total task sets. NP-EDF outperforms SPF because a link does not require to be strictly periodic in NP-EDF schedule. Thus, a link can explore more phasings in NP-EDF scheduler than SPF scheduler. Fig. 5.6 shows the network overhead comparison. We report the network overhead as the percentage of time to transmit the TDMA schedule frame in a beacon period. In Wi-Fi network, the default beacon period is 100ms. We transmit the beacon frame two times, to ensure the management frame is reliably delivered. We observe the network management overhead for NP-EDF is from 2% to 6% as the number of tasks are increased. On the contrary, the network management overhead of SPF is significantly smaller than NP-EDF. SPF reduces the management overhead by 80% compared with NP-EDF, and SPF uses at most 1.2% channel time when the number of tasks is 60.

We use the same task sets to compare the benefit of the overbooking mechanism. We define the schedulability improvement ratio (SIR) as follows:

$$\text{SIR} = \frac{N_B - N_O}{N_O} \cdot 100\% \tag{5.5}$$

where N_B is the number of schedulable task sets with overbooking technique, and N_O is the number of schedulable tasks without overbooking technique. Fig. 5.7 shows the SIR of NP-EDP and SPF scheduler with the overbooking technique. When the number of tasks is low, the SIR is small because the

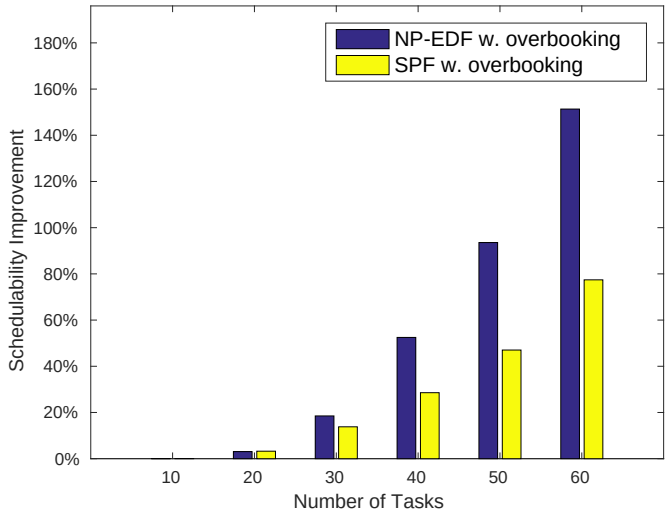
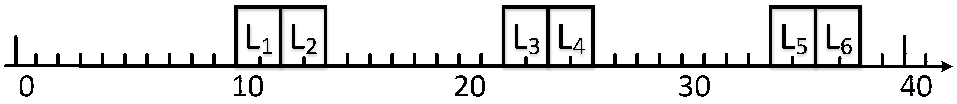


Figure 5.7: Schedulability improvement with overbooking technique.

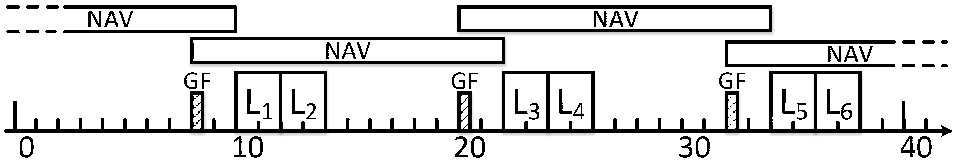
original scheduler can schedule most of the task sets already. The SIR increases as the number of task increases, because as the workload increases, a scheduler with overbooking technique can explore more scheduling opportunities.

5.5.4 Co-existence with Regular Wi-Fi Networks

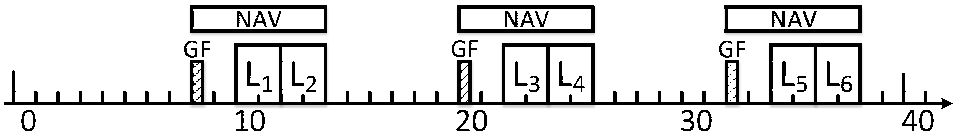
To test the performance of the RT-WiFi network when it coexists with the regular Wi-Fi network, we set up an RT-WiFi network and a regular Wi-Fi network side by side. The RT-WiFi network consists of one RT-WiFi AP and six RT-WiFi stations. We configure a NETGEAR WNDR3700 wireless router as a regular Wi-Fi AP that uses IEEE 802.11a on a 5GHz channel. To create a regular RT-WiFi link, one PC is connected to the router through Ethernet as data receiver and another PC is connected to the router through



(a) A schedule without virtual carrier sensing.



(b) A schedule with aggressive virtual carrier sensing.



(c) A schedule with virtual carrier sensing and bandwidth sharing.

Figure 5.8: Communication schedules of co-existence experiment.

the regular Wi-Fi protocol as data transmitter. Both the RT-WiFi and regular Wi-Fi networks are configured in an office environment using the same 5GHz channel.

There are six links $\{L_i\}_{i=1}^{6}$ in the RT-WiFi network. We configure a link to each station. Because of limited computational power of embedded Galileo board, we set the mini time slot to 256μs. The period of each communication link is 40, and we allocate 2 mini time slot transmission time for each link. The communication schedule is shown in Fig. 5.8a. For this communication schedule, we use iperf to generate the traffic for every communication link. On each RT-WiFi link, we generate a UDP flow with 1450 bytes application layer payload at 1.133Mbps bit rate. To generate interference to the RT-WiFi network, we use iperf to configure another UDP flow with 1450 bytes

application payload at 30Mbps bit rate on the regular Wi-Fi link. For the following experiments, we run each experiment for 10 minutes. We report the packet loss of the RT-WiFi links and the throughput of the regular Wi-Fi link.

Table 5.2: Comparison of different co-existence mechanisms.

	Packet Loss Rate of RT-WiFi links						Regular Wi-Fi link Throughput
	L_1	L_2	L_3	L_4	L_5	$L6$	$L_{\text{Regular Wi-Fi}}$
RT-WiFi baseline	0.06%	0.51%	1.1%	0.92%	1.1%	0.12%	N/A
Regular Wi-Fi baseline	N/A						26.9Mbps
Both traffic (No virtual carrier sensing)	3.7%	6.7%	4.3%	8.7%	4.8%	3.2%	21.6Mbps
Both traffic (With virtual carrier sensing)	0.15%	0.63%	1.2%	0.91%	1.3%	0.32%	Disconnected
Both traffic (With virtual carrier sensing and bandwidth sharing)	0.75%	0.77%	1.8%	1.0%	1.7%	0.32%	13.4Mbps

We summarize the co-existence experiment results in Table 5.2. The first two rows in Table 5.2 show the baseline measurement of the RT-WiFi links and the regular Wi-Fi link respectively. As shown in the third row, if the carrier sensing mechanism is disabled, we observe higher packet loss rate on all the RT-WiFi links. The fourth row shows the network performance when we use the carrier sensing mechanism as shown in Fig. 5.8b. Since we allocate NAV aggressively in this schedule, the packet loss rates for RT-WiFi links are similar to those in the baseline. However, the regular Wi-Fi link is blocked completely. To share unused bandwidth with regular Wi-Fi network, we enable the bandwidth sharing mechanism as shown in Fig. 5.8c. With this communication schedule, we preserve similar packet delivery rate for RT-WiFi links, and let the regular Wi-Fi link retains around half of its baseline bandwidth.

Chapter 6

RT-WiFi Network System Implementation and Case Study

We present the design and implementation of RT-WiFi network management platform in this chapter. In Chapter 3, we propose the MAC layer design of RT-WiFi protocol, and demonstrate that RT-WiFi protocol can support various wireless communication environments. Although the MAC layer of RT-WiFi protocol provides predictable data transmission, it requires great effort for users to deploy an RT-WiFi network because of tedious configuration and lacking proper network management tools. Therefore, in this chapter, based on the RT-WiFi MAC layer design, we propose the RT-WiFi network management platform to serve as an infrastructure to dynamically coordinate the resource allocation and to adjust device configurations. We firstly introduce the background information in Section 6.1. We then present the architecture of the network management platform, and describe the network management protocol in Section 6.2. To demonstrate the benefit of applying RT-WiFi network management platform, we build a case study on the cyber-physical healthcare application to evaluate the system performance in Section 6.3.

6.1 Introduction

In this section, we introduce the design goals of the RT-WiFi network management platform. We design this software architecture with the following design goals:

- Compatibility: The proposed network management platform shall have minimum modification to existing IEEE 802.11 protocol. It is complicated for a regular user to configure a wireless network since there are several network configuration tools for existing IEEE 802.11 networks. For example, hostapd [2] is a software that runs in wireless access point (AP) for configuring wireless AP and severing authentication process. wpa_supplicat [9] is a station-side software that handles the authentication process with the authentication servers. In order for users to take advantage of existing networks configuration, our proposed software architecture reuse the existing tools and we only develop necessary tools for supporting new features of RT-WiFi. Therefore, our prospective users can easily migrate from regular Wi-Fi networks to RT-WiFi networks with minimum effort.

- Modularization: We want to modularize our software in a good fashion so that the network management platform could support different hardware and software architectures. Wireless CPS applications utilize diverse hardware and software platforms to build their systems. To support various hardware and software platforms, the design of RT-WiFi man-

agement platform shall provide clean separation of hardware dependent and hardware independent module, and provide clear modularization of the IEEE 802.11 and RT-WiFi management protocol. Currently, our implementation supports general purpose personal computer platform, and embedded platform such as Intel Galileo development board [5]. Thus, the flexibility of the network management design can be easily ported to serve different wireless CPS applications.

- Open Source: The product of this research will be open source for real-time community's benefit. We have encountered numerous hardware and software issues. In order to solve those issues and make our implementation stable, it takes us years of engineering time. To promote further research on wireless CPS, we contribute our research product to the public domain and hope the implementation of RT-WiFi management platform could help other researchers to develop and investigate their wireless CPS applications.

Based on the aforementioned design goals, we shall present the software architecture on both RT-WiFi AP and station (STA), and describe important software components of network management tools in the following.

6.2 Design and Implementation of Network Management Platform

6.2.1 Network Management Architecture

Fig. 6.1 gives an overview of the software architecture of the RT-WiFi network management platform on both RT-WiFi AP and RT-WiFi STA. This design is developed based on Linux operating system, and IEEE 802.11 compatible hardware. To meet the tight timing constraint, we develop RT-WiFi MAC module and RT-WiFi rate control module in the kernel space. The less time critical components, such as the network manager and station agent, are implemented in the user space in order to easily port and reconfigure to other hardware platforms. To be backward compatible with existing system design, we integrate RT-WiFi management platform with two existing kernel modules, mac80211 and hardware dependent MAC, in the kernel space. Mac80211 is a kernel module that manages MAC sublayer management entity for IEEE 802.11 protocol in Linux, and the hardware dependent MAC kernel module is a device driver for controlling a particular wireless network interface. Based on the proposed RT-WiFi MAC layer design in Chapter 3, we develop RT-WiFi kernel module that supports various configurable options. The RT-WiFi kernel module is further enhanced with generic netlink socket to communicate with the network management software in the user space. For this design, we only modify less than 300 lines of C code in the original Linux kernel that makes minimum changes in the original system.

The are two types of message flows in this network management ar-

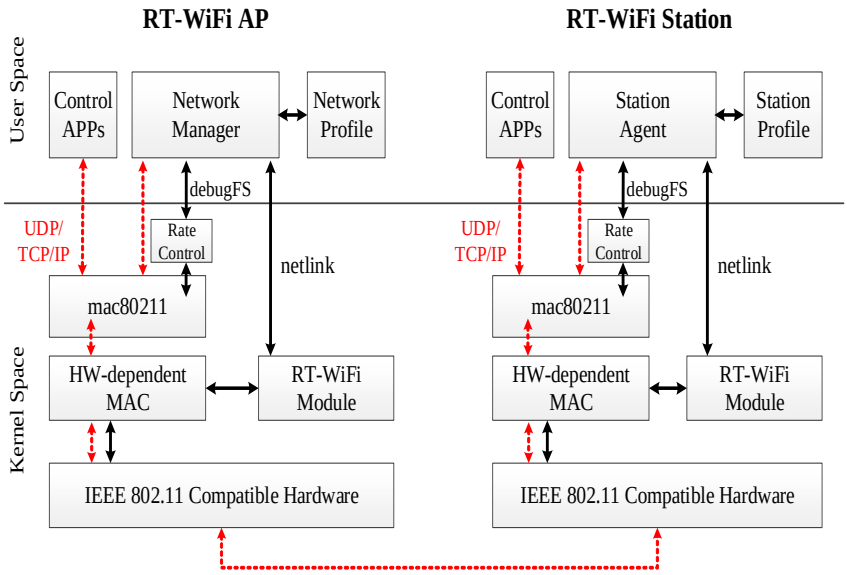


Figure 6.1: An overview of the RT-WiFi software architecture

chitecture. The solid lines in Fig. 6.1 represents the internal messages that are exchanged through internal interface within the RT-WiFi AP or RT-WiFi station. The dashed lines shows the communication messages that are transmitted across different machines. For the internal messages, netlink socket and debugFS are used for communication between user space programs and kernel modules. For messages across different machines, it is transmitted with TCP or UDP protocols. In the following, we shall discuss two key components, network manager and station agent, in the software architecture.

6.2.2 Network Manager

The RT-WiFi network manager is a user space application software that coordinates all the network communication and resource allocation in the

RT-WiFi network. The network manager serves as a general platform that supports different data link layer scheduling algorithms for various networking environments, and uses the network management protocols to be described in Section 6.2.4 to coordinate the message exchanges between RT-WiFi AP and STAs. According to the target application, the application designers specify their network resource requirements in the network management configuration profile located in the RT-WiFi AP.

The configuration profile consists of network-related and AP-related settings. The network-related settings include the reliability requirements, the size of the time slot, the co-existence mode, and the access control list. The AP-related settings include the communication requirements of the broadcast slot. When we launch the RT-WiFi network, we firstly runs the network manager to initialize the RT-WiFi network. The network manager reads the network configuration profile, and use netlink socket to set up the configurable TDMA link layer parameters in RT-WiFi kernel module. After the initialization setup is ready, the network manager waits for the join requests from RT-WiFi stations, and response the joined stations with network configurations.

6.2.3 Station Agent

The station agent is a user space application software which runs in an RT-WiFi station to communicate with RT-WiFi network manager to acquire network resources. To run a station agent, an RT-WiFi user uses the station

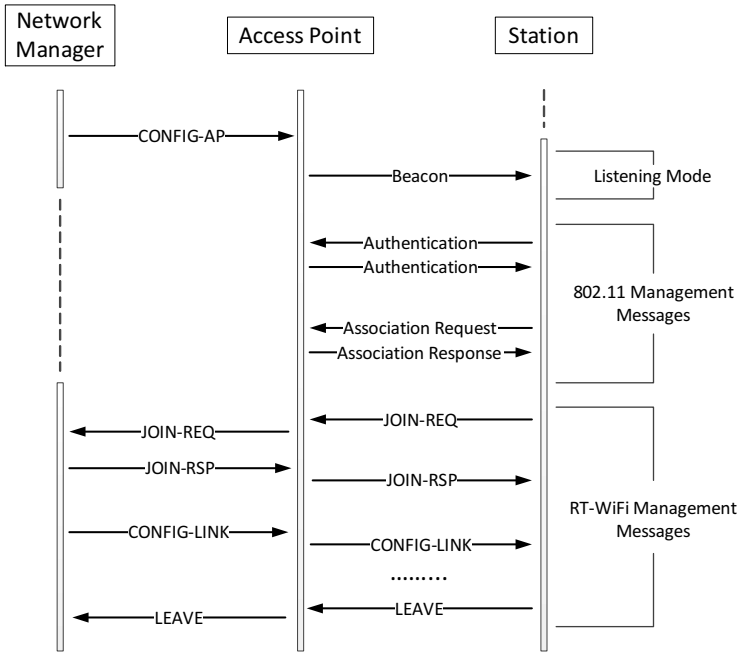


Figure 6.2: An overview of the RT-WiFi management protocol.

configuration profile to configure its station agent. The station configuration profile consists of RT-WiFi network-related setting and its communication resource requirement. The RT-WiFi network-related setting includes the network identification of RT-WiFi AP, and the MAC address of the station. The communication resource requirement specifies the data size and sampling rate of the real-time communication traffic. The station agent reads the station configuration profile, and use the network management protocol in Section 6.2.4 to communicate with RT-WiFi network manager.

6.2.4 RT-WiFi Network Management Protocols

Figure 6.2 illustrates the life cycle of a station in an RT-WiFi network and the exchanged management message sequence. There are two phases in the life cycle of an RT-WiFi network, an initialization phase and an operation phase. In the initialization phase, the network manager sends a CONFIG-AP message to configure the AP according to the configuration profile. After the AP is configured properly, it starts its TDMA MAC layer, broadcasts the TDMA information in the beacon message periodically, and forms an RT-WiFi network. In the operation phase, when an RT-WiFi STA attempts to join the network, it scans the channels for the beacon frame, and then authenticates and associates with the AP through regular IEEE 802.11 management frames in reserved time slots. After the associate process finishes, the RT-WiFi STA sends out a JOIN-REQ message to specify its communication resource requirement, and ask the network manager for its data link layer communication schedule. The network manager executes the implemented scheduling algorithms to compute the communication schedule and replies the STA with a JOIN-RSP message. After the STA receives this information, it starts its TDMA state machine and starts to exchange data messages with the RT-WiFi AP according to the configured communication schedule. Due to the join and leave of other stations in the network, the network manager may need to update the communication schedule in the run time to adapt to network dynamics. A CONFIG-LINK messages will be sent out by the network manager to reflect a communication adjustment to the affected stations. Finally, when

a station wants to leave the network, it will send out a LEAVE message to notify the network manager, and the network manager will reclaim the allocated time slots for this station.

6.3 Case Study - A Mobile Gait Rehabilitation System

To evaluate the benefit of applying RT-WiFi communication platform in wireless cyber-physical applications, we have built a mobile gait rehabilitation system with RT-WiFi communication platform. The mobile gait rehabilitation system is a healthcare cyber-physical system which allows the patients to do rehabilitation at an open environment, such as home or community, instead of at a closed hospital environment.

Fig. 6.3 gives an overview of the mobile gait rehabilitation system, which includes patient side subsystem and therapist side subsystem. The patient side subsystem consists of a local computer and heterogeneous wearable sensing and control devices, including "smart shoes" [48] with embedded air pressure sensors, multiple wireless motion sensors, and wireless assistive robots [47]. The local computer receives data from the sensing devices, processes the data, and sends control signals to the wireless assistive robots. An Internet connection is established between the local computer and the host computer at therapist side, and the local computer periodically reports processed data to the therapist's host computer . On the therapist side, the host computer logs the processed sensing data from the local computer, so that therapist can monitor the rehabilitation progress and make rehabilitation

plan.

RT-WiFi network serves as the wireless communication system between the local computer and the sensing and control devices for improved system mobility and tele-rehabilitation. The local computer serves as RT-WiFi AP and runs control algorithms in the application layer. The sensing and control devices are attached to the RT-WiFi stations depending on their locations. There are two types of wireless communication links in the system. One type of link is used to transmit sensing signals from sensing devices (air pressure sensors and wireless motion sensors) to the control applications in the local computer [14]. Information received on this wireless link is used for health monitoring and rehabilitation strategy design, so the sampling rate can be chosen from $100Hz$ to $1kHz$ depending on the high-level decision making algorithms ($100Hz$ is chosen in the current design). The other type of wireless link is for controlling the wireless assistive robots, which require at least $1kHz$ sampling rate for precise motion control. More details of the system design can be found in [82].

6.3.1 System Integration

We take the smart shoe as an example to show how we integrate the sensing and control devices into the rehabilitation system through RT-WiFi communication platform. Smart shoe is a human gait detecting device, which has been developed for health monitoring from our previous research.

Our mobile gait rehabilitation system periodically requests sensing sig-

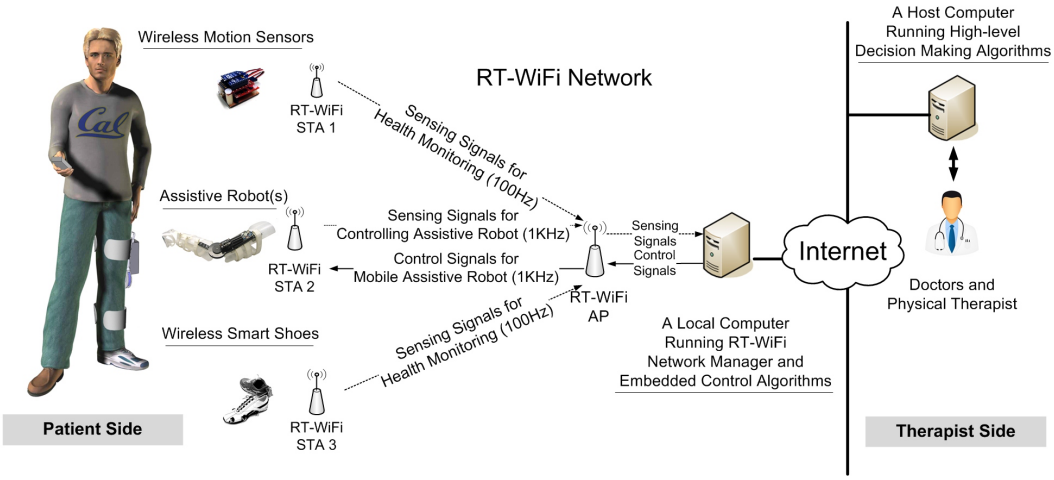


Figure 6.3: An overview of the mobile gait rehabilitation system

nals from air pressure sensors for abnormal gait detection. There are four air pressure sensors embedded in each shoe, and sensing signal from each sensor is encapsulated as an 8-byte packet. Including an 8-byte time index, we have 72 bytes of data in total for two shoes to be transmitted in each time step. Fig. 6.4 shows how we integrated smart shoes into the RT-WiFi network. The real-time sensor data were first collected from an analog-input module NI 9221 [11] on an NI 9116 compactRIO chassis, and then sent through a UDP socket from an Ethernet port of the NI 9022 real-time controller on the compactRIO chassis to a RT-WiFi station. The RT-WiFi station then forwarded the sensor data through the RT-WiFi wireless communication link to the RT-WiFi AP on which the controller was running.

To evaluate the performance of the network-based rehabilitation system, we measure the application layer latency between the smart shoes and the

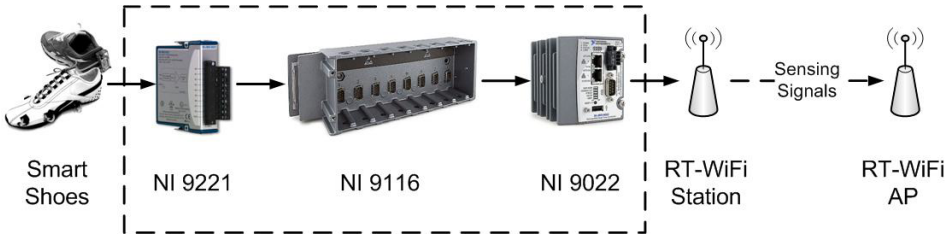


Figure 6.4: Integration of smart shoes with a RT-WiFi station

RT-WiFi AP, which relies on a precise system clock synchronization between them. To achieve this synchronization, we utilize IEEE 1588 Precision Time Protocol (PTP) [34] by using its software implementation PTP daemon [12], and using a dedicate PCI Express Ethernet link for PTP. The synchronization error between the two system clocks was measured to be under $40\mu s$. This was precise enough to measure the end-to-end application layer latency and compare the control performance between RT-WiFi and regular Wi-Fi.

6.3.2 Emulation of a Wireless Control System

To better demonstrate the performance of the RT-WiFi wireless communication protocol in a real-life control application, we ran numerous simulations based on the data traces we collected from the smart shoes hardware. Note that in the real control system, there should be two links for transmitting both sensing and control signals as shown in Fig. 6.3. As the first step of wireless control system integration, we use data from the smart shoes to acquire the network dynamics (latency and packet loss ratio), and then ran some simulations to evaluate the performance of the wireless control system before we integrate the robotic hardware.

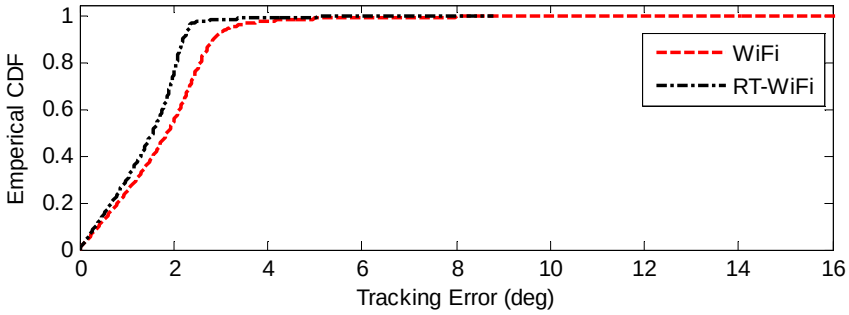
The model of our robotic device in this networked rehabilitation system was used for simulation. We ran both regular Wi-Fi and RT-WiFi network at $1kHz$ for 60 seconds in an indoor office environment. At each time step, we recorded whether the sensing packet was successfully transmitted to the wireless AP. If so, we recorded the latency between the application layer of wireless station and wireless AP. In this simulation, the control signal was implemented every $1ms$, and a sensing packet would be ignored if its latency was larger than $1ms$. To compare the performance of wireless network for control system applications, we define the effective packet loss ratio (EPLR) as follows:

$$EPLR = \frac{N_l + N_d}{N} \times 100\%, \tag{6.1}$$

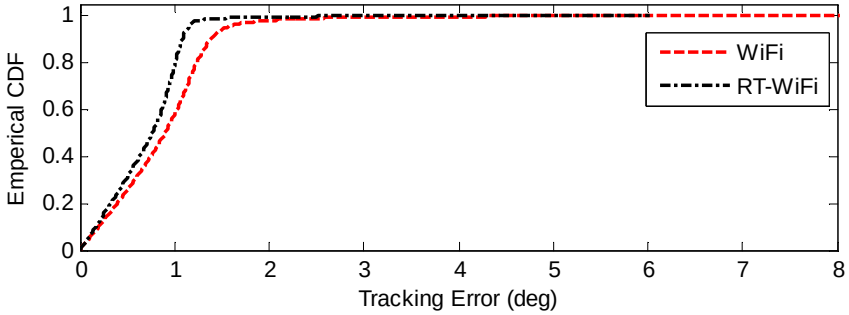
where N_l is the number of lost packets, N_d is the number of packets with latency longer than the sampling interval, and N is the total number of packets transmitted over a certain period.

A proportional-derivative (PD) controller was implemented as the main controller, and the wireless control system was set to track a sinusoidal signal with the magnitude of one radian. Simulations were conducted with different controller gains and reference frequencies to test the effectiveness of network dynamics. The root-mean-square (RMS) error are given in Table 6.1. ECDFs of tracking errors are given in Fig. 6.5 and Fig. 6.6.

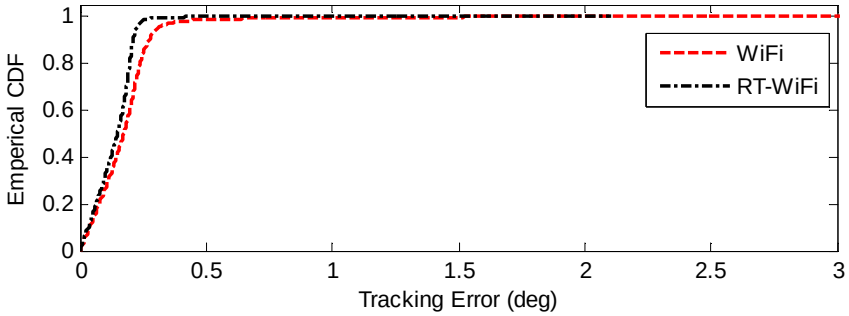
Based on our simulation, the EPLR for regular Wi-Fi is 26.07%, and the EPLR for RT-WiFi is 9.5%, which is 63.56% lower than regular Wi-Fi.



(a) ECDF of tracking errors with $K_p = 0.5, K_d = 0.005$

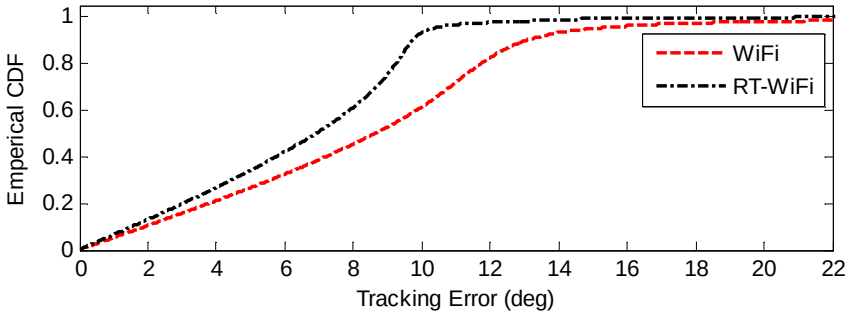


(b) ECDF of tracking errors with $K_p = 1, K_d = 0.01$

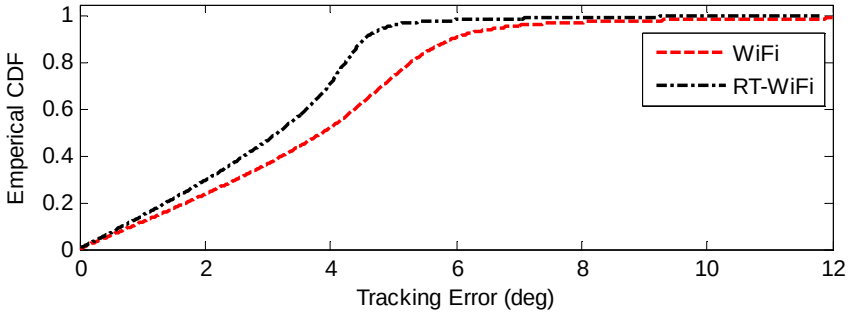


(c) ECDF of tracking errors with $K_p = 5, K_d = 0.05$

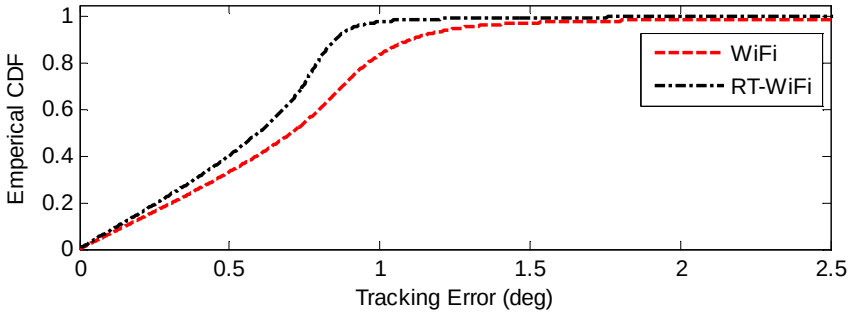
Figure 6.5: ECDFs of tracking errors with $1Hz$ reference



(a) ECDF of tracking errors with $K_p = 0.5, K_d = 0.005$



(b) ECDF of tracking errors with $K_p = 1, K_d = 0.01$



(c) ECDF of tracking errors with $K_p = 5, K_d = 0.05$

Figure 6.6: ECDFs of tracking errors with $2Hz$ reference

	Controller Gain		RT-WiFi	Wi-Fi	Improvement
	K_p	K_d			
RMS Error (deg) reference frequency= $1Hz$	0.5	0.005	1.633	2.097	22.13%
	1	0.01	0.796	1.037	23.24%
	5	0.05	0.160	0.257	37.74%
RMS Error (deg) reference frequency= $2Hz$	0.5	0.005	7.236	9.872	26.70%
	1	0.01	3.323	4.526	26.58%
	5	0.05	0.636	1.140	44.21%

Table 6.1: RMS tracking errors in simulations

As we can see from Table 6.1 that compared with regular Wi-Fi, RT-WiFi consistently yields much smaller RMS tracking errors under different settings of the controller gains. This demonstrates the significant improvement on control performance compared with regular Wi-Fi. From Figs. 6.6, we can see that the probability to achieve small tracking errors is always higher for RT-WiFi than regular Wi-Fi. Moreover, the system is more sensitive to packet loss when the controller gain is smaller, and larger tracking errors occur when we ask the system to track faster reference signals. In clinic test, to protect patients from injury, we need to make sure that the position tracking error is no larger than 10 degrees at all times. Due to the limitation of actuator power, we also prefer small control gains (e.g., $K_p = 0.5$, $K_d = 0.005$). In this setting, only RT-WiFi can meet the control application requirements, while regular Wi-Fi would incur health risks to the patients.

Chapter 7

Conclusion and Future Work

7.1 Summary

Applying real-time wireless technology to CPSs has drawn a lot of attentions in these years. In this dissertation, we aim to provide a real-time wireless communication platform for emerging CPSs. We firstly build a flexible TDMA MAC layer protocol for the RT-WiFi communication system. We then present the design principles and implementation details of the RT-WiFi protocol to support real-time predictable data delivery with high sampling rate. It incorporates configurable components for adjusting design trade-offs including sampling rate, real-time performance, communication reliability, and compatibility to existing Wi-Fi networks. RT-WiFi MAC layer protocol can be implemented on commercial off-the-shelve hardware and supports up to $6kHz$ sampling rate with IEEE 802.11 a/g physical layer.

Based on the proposed TDMA MAC layer, we present network management techniques to schedule network resource and provide reliable communication in noisy environments. We propose efficient network management algorithms for controlling jitter in wireless CPSs. We design a harmonic-chain based jitter-free (HCJF) scheduler to select the sampling period for each com-

munication task for eliminating the transmission jitter. Considering network dynamics, we further introduce an efficient $\mathcal{S}$-tree structure to represent the data link layer communication schedule in the network, and present an efficient algorithm for dynamic network resource allocation while minimizing the network adjustment overhead.

To provide reliable communication, we consider the interference from both Wi-Fi and non-Wi-Fi interference sources, and develop various solutions to support real-time reliable wireless communication for the RT-WiFi network. For non-Wi-Fi interference, we apply rate adaptation and retransmission mechanisms to the network manager design that dynamically constructs and distributes reliable communication schedules to the network. To build reliable communication schedules, we present an optimal rate control algorithm RTRA, a communication link scheduler SPF that has low network management overhead, and the "overbooking" technique that further improves the schedulability of the network link scheduler. For regular Wi-Fi based interference, we propose a virtual carrier sensing based approach that prioritizes transmission of RT-WiFi traffic and shares unused channel time to regular Wi-Fi network.

Finally, to demonstrate the effectiveness of the proposed RT-WiFi system, we build the network management platform and a case study on healthcare CPS application. The development of RT-WiFi management platform aims to meet the stringent timing requirement and to provide a modularization design that it can be easily adapted to various CPSs. We also integrate

RT-WiFi communication system into a mobile gait rehabilitation system. Results from extensive experiments validate our design and demonstrate the advantage of RT-WiFi over regular Wi-Fi. Our results lead to more effective controller design with significantly better control performance.

7.2 Future Research

7.2.1 Multi-cluster RT-WiFi Network

We shall consider how to deploy RT-WiFi networks in a large area that requires more than one RT-WiFi cluster. Current RT-WiFi infrastructure only supports a single RT-WiFi cluster that consists of one AP and multiple stations. To cover an area of massive deployment of sensors and actuators in industrial environments, we shall expand our system architecture to support multiple RT-WiFi clusters. The research issues include how to coordinate the inter-cluster interference across multiple RT-WiFi clusters, how to support end-to-end performance guarantees across RT-WiFi clusters, and how to distribute network management functions to lower the burden of a single network manager and to avoid single point of failure.

7.2.2 Failure Semantics for CPSs

We shall investigate the failure semantics for wireless CPSs. The main focus of this dissertation is on providing real-time and reliable wireless communication systems for CPSs. However, we expect that under the extreme interference that might be encountered in some industrial settings, not only

the communication subsystem but also the application-layer failure schematics must be exploited to meet the application-specific performance challenge. The research issues include how to specify failure schematics for wireless CPSs and how to design an interface for the communication subsystem to notify the application layer when a failure is occurred.

Appendices

Appendix A

Notational Conventions

[**L**] a real-time communication link

[**P**] period of a communication link

[**D**] deadline of a communication link

[**c**] the transmission time of a communication link

[ϕ] phasing of a communication link

[**B**] number of bytes of data to be transmitted for a communication link per period

[**Q**] expected data delivery rate of a communication link

[**T**] a retry chain of a communication link

[**X(T)**] transmission time of a retry chain

[**E(T)**] expected data delivery rate of a retry chain

[**R**] available data rates in the RT-WiFi system

[**O**] transmission overhead in the RT-WiFi system

Appendix B

Acronyms

[**AP**] Access Point

[**CPS**] Cyber-Physical System

[**CSMA/CA**] Carrier Sense Multiple Access with Collision Avoidance

[**EDF**] Earliest Deadline First

[**HCJF**] Harmonic-Chain Jitter-Free

[**MAC**] Media Access Control

[**NP-EDF**] Non-Preemptive Earliest Deadline First

[**RM**] Rate Monotonic

[**RTRA**] Real-Time Rate Adaptation

[**SPF**] Shortest Period First

[**STA**] Station

[**TDMA**] Time Division Multiple Access

Bibliography

[1] Bluetooth. http://www.bluetooth.com.

[2] hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. https://w1.fi/hostapd/.

[3] IEEE 802.11 working group. http://www.ieee802.org/11/.

[4] IEEE 802.15 WPAN task group 4. http://www.ieee802.org/15/pub/TG4.html.

[5] Intel Galileo Development Board. http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-overview.html.

[6] Iperf. http://iperf.sourceforge.net/.

[7] ISA100. http://www.isa.org/isa100.

[8] Linux Compat-wireless Driver. http://wireless.kernel.org/.

[9] Linux wpa/wpa2/IEEE 802.1x supplicant. http://w1.fi/wpa_supplicant/.

[10] Minstrel. https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel.

[11] National Instruments. http://www.ni.com/compactrio/.

[12] Precision Time Protocol Daemon. `http://sourceforge.net/projects/ptpd/`.

[13] Zigbee alliance. `http://www.zigbee.org`.

[14] J. Bae, K. Haninger, X. Wai, D.and Garcia, and M. Tomizuka. A network-based monitoring system for rehabilitation. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 232–237, 2012.

[15] P. Bahl, R. Chandra, and J. Dunagan. SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 216–230. ACM, 2004.

[16] A. Bar-Noy, V. Dreizin, and B. Patt-Shamir. Efficient periodic scheduling by trees. In *IEEE International Conference on Computer Communications*, volume 2, pages 791–800, 2002.

[17] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari. Scheduling periodic task systems to minimize output jitter. In *Real-Time Computing Systems and Applications (RTCSA)*, pages 62–69, 1999.

[18] V. Bonifaci and A. Wiesex A. Marchetti-Spaccamelay, N. Megowz. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 236–245, 2013.

[19] Yang Cai and M. C. Kong. Nonpreemptive scheduling of periodic tasks in uni- and multiprocessor systems. *Algorithmica*, 15(6):572–599, 1996.

[20] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 298–307, 2003.

[21] G. Cecchetti and A. L. Ruscelli. Performance evaluation of real-time schedulers for HCCA function in IEEE 802.11e wireless networks. In *Proceedings of the 4th ACM symposium on QoS and security for wireless and mobile networks*, pages 1–8, 2008.

[22] T. Chantem, X. Wang, M.D. Lemmon, and X.S. Hu. Period and deadline selection for schedulability in real-time systems. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS)*, pages 168–177, 2008.

[23] D. Chen, A.K. Mok, and T.-W. Kuo. Utilization bound revisited. *IEEE Transactions on Computers*, 52(3):351–361, 2003.

[24] O. Chipara, C. Wu, C. Lu, and W. Griswold. Interference-aware real-time flow scheduling for wireless sensor networks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 67–77, July 2011.

[25] R Costa, P Portugal, R Moraes, and F Vasques. An admission control mechanism to handle real-time traffic in ieee 802.11 networks in open com-

munication environments. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 63–66, 2012.

[26] R. Costa, P. Portugal, F. Vasques, and R. Moraes. A TDMA-based mechanism for real-time communication in IEEE 802.11 e networks. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–9. IEEE, 2010.

[27] Robson Costa, Paulo Portugal, Francisco Vasques, Carlos Montez, and Ricardo Moraes. Limitations of the ieee 802.11 dcf, pcf, edca and hcca to handle real-time traffic. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pages 931–936. IEEE, 2015.

[28] Robson Costa, Paulo Portugal, Francisco Vasques, Ricardo Moraes, and Ricardo Felipe Custódio. A coordination layer to handle real-time communication in wi-fi networks with uncontrolled traffic sources. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 263–266. IEEE, 2011.

[29] A. Dhekne, N. Uchat, and B. Raman. Implementation and evaluation of a TDMA MAC for WiFi-based rural mesh networks. In *ACM Workshop on Networked Systems for Developing Regions*, 2009.

[30] M. Di Natale and J. A. Stankovic. Scheduling distributed real-time tasks with minimum jitter. *IEEE Transactions on Computers*, 49(4):303–316, 2000.

[31] P. Djukic and P. Mohapatra. Soft-TDMAC: A software TDMA-based MAC over commodity 802.11 hardware. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1836–1844, 2009.

[32] Petar Djukic and Prasant Mohapatra. Soft-tdmac: a software-based 802.11 overlay tdma mac with microsecond synchronization. *Mobile Computing, IEEE Transactions on*, 11(3):478–491, 2012.

[33] L. Dong, R. Melhem, and D. Mosse. Effect of scheduling jitter on end-to-end delay in TDMA protocols. In *Real-Time Computing Systems and Applications (RTCSA)*, pages 223–230, 2000.

[34] J.C. Eidson. *Measurement, control, and communication using IEEE 1588*. Springer, 2010.

[35] Friedrich Eisenbrand, Nicolai Hähnle, Martin Niemeier, Martin Skutella, José Verschae, and Andreas Wiese. Scheduling periodic tasks in a hard real-time environment. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010*, pages 299–311, 2010.

[36] C. Ekelin. Clairvoyant non-preemptive edf scheduling. In *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, pages 7 pp.–32, 2006.

[37] T. Erlebach, R. Jacob, M. Mihalak, M. Nunkesser, G. Szabo, and P. Widmayer. An algorithmic view on OVSF code assignment. *Algorithmica*,

47(3):269–298, 2007.

[38] V. Gabale, B. Raman, K. Chebrolu, and P. Kulkarni. LiT MAC: addressing the challenges of effective voice communication in a low cost, low power wireless mesh network. In *ACM Symposium on Computing for Development*, page 5, 2010.

[39] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[40] S. Han, A. K. Mok, J. Meng, Y.-H. Wei, P.-C. Huang, Q. Leng, X. Zhu, L. Sentis, K. S. Kim, and R. Miikkulainen. Architecture of a cyber-physical avatar. In *International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

[41] S. Han, X. Zhu, D. Chen, A. K. Mok, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 3–12, 2011.

[42] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, MobiCom '01, pages 236–251, New York, NY, USA, 2001. ACM.

[43] I. H. Hou, V. Borkar, and P. R. Kumar. A theory of qos for wireless. In *IEEE INFOCOM*, pages 486–494, April 2009.

[44] I. H. Hou and P. R. Kumar. Scheduling heterogeneous real-time traffic over fading wireless channels. In *IEEE INFOCOM*, pages 1–9, March 2010.

[45] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of period and sporadic tasks. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 129–139, Dec 1991.

[46] Glenn Judd, Xiaohui Wang, and Peter Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, MobiSys '08, pages 118–131, 2008.

[47] K. Kong, J. Bae, and M. Tomizuka. A compact rotary series elastic actuator for human assistive systems. *IEEE/ASME Transactions on Mechatronics*, 17(2):288–297, 2012.

[48] K. Kong and M. Tomizuka. A gait monitoring system based on air pressure sensors embedded in a shoe. *IEEE/ASME Transactions on Mechatronics*, 14(3):358–370, 2009.

[49] Jan Korst, Emile Aarts, and Jan Karel Lenstra. Scheduling periodic tasks. *INFORMS journal on Computing*, 8(4):428–435, 1996.

[50] D. Koutsonikolas, T. Salonidis, H. Lundgren, P. LeGuyadec, Y. C. Hu, and I. Sheriff. TDM MAC protocol design and implementation for wireless mesh networks. In *ACM CoNEXT Conference*, page 28, 2008.

[51] T.-W. Kuo and A. K. Mok. Incremental reconfiguration and load adjustment in adaptive real time systems. *IEEE Transactions on Computers*, 46(12):1313–1324, 1997.

[52] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 126–134, 2004.

[53] Edward Lee et al. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

[54] Q. Leng, Y.-H. Wei, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka. Improving control performance by minimizing jitter in RT-WiFi networks. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 63–73, 2014.

[55] B. Li, Z. Sun, K. Mechitov, G. Hackmann, C. Lu, S. J. Dyke, G. Agha, and B. F. Spencer. Realistic case studies of wireless structural control. In *International Conference on Cyber-Physical Systems (ICCPS)*, pages 179–188, 2013.

[56] C. Y. Li, C. Peng, S. Lu, X. Wang, and R. Chandra. Latency-aware rate adaptation in 802.11n home networks. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1293–1301, April 2015.

[57] K.-J. Lin and A. Herkert. Jitter control in time-triggered systems. In *IEEE Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, volume 1, pages 451–459, 1996.

[58] M. Marouf and Y. Sorel. Scheduling non-preemptive hard real-time tasks with strict periods. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8, 2011.

[59] P. Marti, J. M. Fuertes, G. Fohler, and K. Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 91–100, 2002.

[60] A. K. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. Algorithms and complexity of the periodic maintenance problem. In *Microprocessing and Microprogramming*, volume 27, pages 657–664, 1989.

[61] R. Moraes, F. Vasques, P. Portugal, and J. A. Fonseca. VTP-CSMA: A virtual token passing approach for real-time communication in IEEE 802.11 wireless networks. *IEEE Transactions on Industrial Informatics*, 3(3):215–224, 2007.

[62] M. Nasri and G. Fohler. An efficient method for assigning harmonic periods to hard real-time tasks with period ranges. In *2015 27th Euromicro Conference on Real-Time Systems*, pages 149–159, July 2015.

[63] Mitra Nasri and Mehdi Kargahi. Precautious-RM: a predictable non-preemptive scheduling algorithm for harmonic tasks. *Real-Time Systems*,

50(4):548–584, 2014.

[64] D. Panigrahi and B. Raman. TDMA scheduling in long-distance WiFi networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 2931–2935. IEEE, 2009.

[65] Rabin Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. Wildnet: Design and implementation of high performancewifi based long distance networks. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, pages 7–7. USENIX Association, 2007.

[66] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.

[67] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale. PIP: a connection-oriented, multi-hop, multi-channel TDMA-based MAC for high throughput bulk transfer. In *ACM Conference on Embedded Networked Sensor Systems*, pages 15–28, 2010.

[68] Bhaskaran Raman and Kameswari Chebrolu. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, pages 156–169, New York, NY, USA, 2005. ACM.

[69] A. Rao and I. Stoica. An overlay MAC layer for 802.11 networks. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148. ACM, 2005.

[70] A. Saifullah, Y. Xu, C. Lu, and Y. Chen. End-to-end delay analysis for fixed priority scheduling in wirelessHART networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 13–22, 2011.

[71] F Santos, L Almeida, P Pedreiras, L.S. Lopes, and T Facchinetti. An adaptive tdma protocol for soft real-time wireless communication among mobile autonomous agents. In *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS*, volume 2004, pages 657–665, 2004.

[72] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. Accurate: Constellation based rate estimation in wireless networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 12–12, 2010.

[73] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 13–21, 1996.

[74] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. WirelessHART: Applying wireless technology in real-time industrial pro-

cess control. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 377–386, 2008.

[75] Wim Torfs and Chris Blondia. TDMA on commercial of-the-shelf hardware: Fact and fiction revealed. *AEU - International Journal of Electronics and Communications*, 69(5):800 – 813, 2015.

[76] D. Vassis, G. Kormentzas, A. Rouskas, and I. Maglogiannis. The IEEE 802.11g standard for high data rate WLANs. *Network, IEEE*, 19(3):21–26, 2005.

[77] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka. RT-WiFi: Real-time high-speed communication protocol for wireless cyber-physical control applications. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 140–149, 2013.

[78] Yi-Hung Wei, Quan Leng, Song Han, Aloysius K Mok, Wenlong Zhang, Masayoshi Tomizuka, Tianji Li, David Malone, and Douglas J Leith. RT-WiFi: real-time high speed communication protocol for wireless control systems. *SIGBED Review*, 10(2):28, 2013.

[79] Yi-Hung Wei, Quan Leng, and Aloysius K. Mok. Enhancing reliability in RT-WiFi network. Technical report, The University of Texas at Austin, Department of Computer Science. Report# TR-16-11 (regular tech report), 2016.

[80] D. B. West. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[81] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 146–157, 2006.

[82] Wenlong Zhang, Xiuming Zhu, Song Han, Nancy Byl, Aloysius K Mok, and Masayoshi Tomizuka. Design of a network-based mobile gait rehabilitation system. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1773–1778, 2012.

[83] X. Zhu, S. Han, P.-C. Huang, A. K. Mok, and D. Chen. Mbstar: A real-time communication protocol for wireless body area networks. In *IEEE Euromicro Conference on Real-Time Systems (ECRTS)*, pages 57–66, 2011.

[84] X. Zhu, P.-C. Huang, S. Han, A.K. Mok, D. Chen, and M. Nixon. Minmax: A sampling interval control algorithm for process control systems. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 68–77, 2012.

Vita

Yi-Hung Wei was born in HsinChu City, Taiwan. He received the Bachelor of Science degree in the Department of Computer Science from National Tsing Hua University in 2007 and the Master of Science degree in the Department of Computer Science and Information Engineering from National Taiwan University in 2009. Before he started his Ph.D. study at the University of Texas at Austin, he served as an information officer for mandatory service in R.O.C. Navy for one year. During his Ph.D. journey, he has been interned at VMWare, Inc. and Google, Inc. in the summers of 2012 and 2013. He is the recipient of the best paper award in 2013 IEEE Real-Time Systems Symposium (RTSS).

E-mail address: yihung.wei@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.